Visual Studio

Windows 8.1
ユーザー エクスペリエンス
ガイドライン

更新 2014 年 6 月

この文書が翻訳版の場合、オリジナル版と内容に違いが認められた場合にはオリジナル版に従うものとします。記載された情報は発行日時点のものです。

この文書に記載されている URL や参照しているウェブサイトを含む情報や見解は、予告なく変更される場合があます。断りがない限り、例として記載されている企業、組織、製品、ドメイン名、メールアドレス、ロゴ、人名、場所、事象は架空のもので、実在の企業、組織、製品、ドメイン名、メールアドレス、ロゴ、人名、場所、事象とは関連性がなく、また意図したり暗示するものではありません。利用者はすべての適用される著作権法にした従う責任を負うものとします。著作権により権利の制限がない場合は、Microsoft Corporation による明記された許諾がない限り、この文書のいずれの部分も複製、保管、復旧用システムへの取り入れ、いかなる方法 (電子的、機械的、複写式、記録、その他) もしくはいかなる目的で任意の形式への送信をすることができません。

Microsoft は、この文書にある事柄について特許、特許出願中、商標、著作権、もしくはその他の知的財産の権利を保有する場合があります。Microsoft からの文書による使用許諾契約に明示的に示された場合を除き、この文書の提供はそれらの特許、商標、著作権もしくはその他の知的財産のいかなる許諾を与えるものではありません。

Microsoft は、米国における Microsoft Corporation および、またはその他の国における登録商標もしくは商標です。

実在する会社の名称と製品はそれぞれの所有者に帰属する場合があります。

Microsoft

目次

はじめに

Windows 8 **プラットフォームとは**

開発者にとって、Windows 8 は、生産性や創造性の向上、そして娯楽のために Windows を毎日使う世界中のたくさんの人々との距離を縮めるすばらしいプラットフォームとなります。また、Windows 8 は、世界規模の顧客にアプローチするためにかつてない手段と機会を提供します。Windows 8 は開発者が使用できる最上位のプラットフォーム機会であり、すばらしい Windows ストア アプリをたくさんの潜在的なユーザーに提供できるビジネスチャンスといえます。

アプリは Windows 8 エクスペリエンスの中心に位置付けられています。アプリはアクティビティやコンテンツによって生命を吹き込まれます。ユーザーは全画面表示の Windows ストア アプリに夢中になって、オペレーティング システムではなくコンテンツに集中できます。

Windows 8 プラットフォームは、生産性および流動性に適した「妥協のない」エクスペリエンスを実現するという約束を果たしています。それはコンシューマーにとっては楽しく、企業にとっては容易に管理できるものです。Windows 8 プラットフォームでは、モバイル デバイスまたはデスクトップ デバイスでのコンテンツの作成に関するガイドラインを定めると同時に、柔軟性と独自の機能を提供しています。

Windows 8.1 は、Windows 8 の基盤の上に構築されており、次世代の革新的なアプリを実現します。Windows 8.1 では、引き続き主役はアプリであり、またシステムを操作する新しい方法が導入されています。 Build 2014 で、ユニバーサル Windows アプリ を発表し、Windows Phone アプリも Windows 8 の基盤の上で構築できるようになりました。Windows ストアでは、優れたマーチャンダイジングとアプリの検索のしやすさにより、優れた収益機会を引き続き提供します。もちろん、Windows では、プログラミング言語 (C#、C++、JavaScript、または VB)、プレゼンテーション テクノロジ (XAML、HTML、または DirectX) の独自の選択範囲を開発者に提供すると共に、Windows ストアによってビジネス モデルを選択できます。Windows 8 プラットフォームを活用したアプリを作成するには、Windows 8.1 が提供するエクスペリンスを活用するためにも デザイン ガイドラインを知ることが重要になります。本ホワイトペーパーでは、エクスペリエンスをデザインするために必要な、基本的な事項を説明します。

デザイン

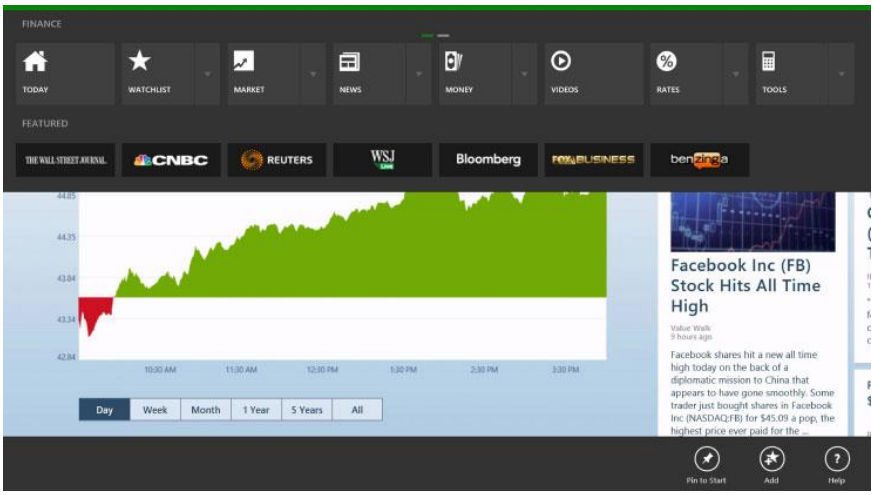
Windows **らしさ**

Windows 8 プラットフォームは、ユーザーに一貫したエクスペリエンスを提供するために共通したデザイン部品を提供しています。ユーザーに Windows 8 と共通したエクスペリエンスを提供するために、Windows らしさを活用することを検討しましょう。

モダン デザイン

Windows は「真のデジタル化」原則を活用しながらモダン デザインをリードしています。そして、スイス スタイルと、地下鉄の案内標識などからインスピレーションを得ています。

エッジ（端）



Windows には、デザイン ツールの 1 つとしてエッジがあります。エッジを効果的に使ってアプリのシナリオにフォーカスし、コンテンツが隠れないようにしましょう。

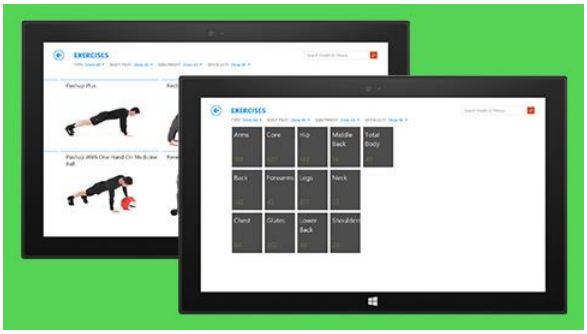
ライブタイル

ライブ タイルは、スタート画面のタイルに直接送られる通知を通じてアプリの使用を促すユニークなツールです。

スプラッシュ スクリーン

スプラッシュ スクリーンは、アプリのエクスペリエンスの最初を飾る、全画面に一瞬だけ表示されるブランド要素です。

セマンティック ズーム



セマンティック ズームは光学式ズームよりも強力です。コンテンツのパン、異なるセクションへのジャンプ、並べ替えのナビゲーションを迅速に行うことができます。

チャームとコントラクト

Windows の核となるのは、すべてのアプリがコントラクトによってチャーム (共有、デバイスなど) に公開する統一的な OS との統合メカニズムです。

モダン デザイン

Microsoft は長らく、多くの分野と製品でテクノロジ リーダーでした。最近になって、よりデザインを中心とする変更に着手しました。こうした変更は、周囲を巻き込むようなワクワクするアイディアを通じて行われ、強力な基盤と優れた手法を提供することで、その他の手法を古臭く感じさせることになります。

モダン デザインの原点

モダン デザインを「フラット デザイン」と呼びたくなるかもしれませんが、それだけではモダン デザインが持つ奥行き、特徴、背景などの多くのことを見落とすことになります。

- The Bauhaus: 1919 年に創設されたこの象徴的なデザインを専門とする学校は、コア機能へとそぎ落とす単純化、不用な冗長性の排除、簡略化による美の尊重など、確固としたデザイン思想を奨励してきました。
- 国際タイポグラフィー様式 (スイス スタイル): スイス スタイルにより、何よりもまず大胆かつクリーンで、美しい書体による Microsoft のデザインの着想を得ています。このスタイルでは、中心的なテーマとしてクリアであること、読みやすいこと、客観的であることを奨励しています。このスタイルから、タイポグラフィーおよび視覚のいずれにおいても、グリッドの必要性を解消し、原則に沿った美しい方法で非対称的なレイアウトを採用しました。

- アニメーションのデザイン: アニメーションは、エクスペリエンスに活気を与え、洗練された印象をもたらします。弊社は、映画の美しいタイトル文字を作成するために書体やデザイン、そしてアニメーションを活用した先駆者である Saul Bass などからインスピレーションを得ました。またアニメーションは、どのように処理が実行されているかを示すために役立ちます。優れたアニメーション デザインは、楽しみを広げるだけでなく、繊細で直感的なデザインにより説明とトレーニングを兼ねることができます。

モダン デザインを基準にして、Microsoft 設計原則を定義しました。

Microsoft の設計原則

優れた Windows ストア アプリを作るための 5 つの原則を次に示します。アプリの計画を立てるときはこの原則を参考にし、設計と開発で何かの選択をするときには常にこの原則に従ってください。

作品へのこだわりを示す

すべての段階で完全かつ洗練されたエクスペリエンスを作り上げます。多くのユーザーの目に触れることが多い細部に時間と手間をかけます。

- 細部までこだわります。
- 安全で信頼できるアプリを作成します。

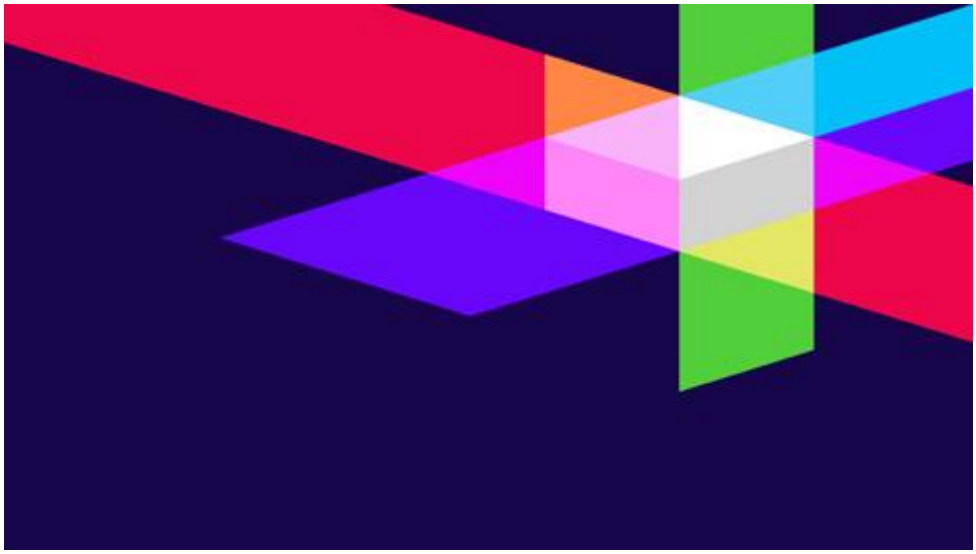
- バランス、対称性、階層を使います。
- アプリをグリッドに合わせて配置します。グリッドは、アプリの新しいレイアウトです。
- 障碍を持つ人も含め、できるだけ多くの人がアプリを利用できるようにします。

軽快かつ柔軟

ユーザーが直接コンテンツを操作できるようにします。相応のエネルギーを注いで、ユーザーの操作にすばやく応答します。意味のあるアニメーションを使って操作に連続性とストーリー性を持たせることで、生き生きとしたエクスペリエンスを提供します。

- ユーザー操作にすばやく応答し、次の操作に備えるようにします。
- タッチ操作や直接的な操作に対応します。
- アニメーションでユーザーを楽しませます。
- 前後の操作へスムーズに接続します。

真のデジタル化を心がける

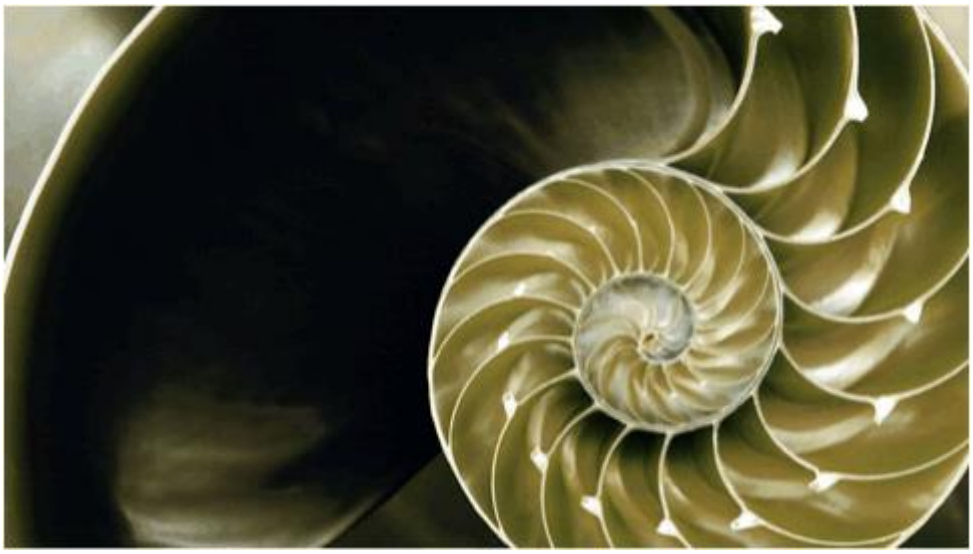


ハードウェアとソフトウェアの機能を明確に示します。デジタル メディアの利点を最大限に活用します。物理的な境界を取り除いて、現実の世界より効率的で容易なエクスペリエンスを実現します。"真のデジタル化" とは、アプリが画面上のピクセルにすぎないという事実を踏まえることです。つまり、現実世界の限界を超えた色と画像を使ったデザインを作ることです。

- ダイナミックで生き生きとしたコミュニケーションを促します。
- 美しいタイポグラフィを使います。
- 大胆で力強い色を使います。
- クラウドに接続して、ユーザーのつながりを実現します。

注: ユーザーは、テクノロジとデジタル操作に精通するようになり、より強力でスケーラブルなデザイン言語に取り組む準備が整ってきました。これが本当のデジタル原則の根拠です。可能な場合は、直感を犠牲にすることなくインターフェイスをそぎ落とすことをお勧めします。

より少ない要素でより大きな効果を上げる

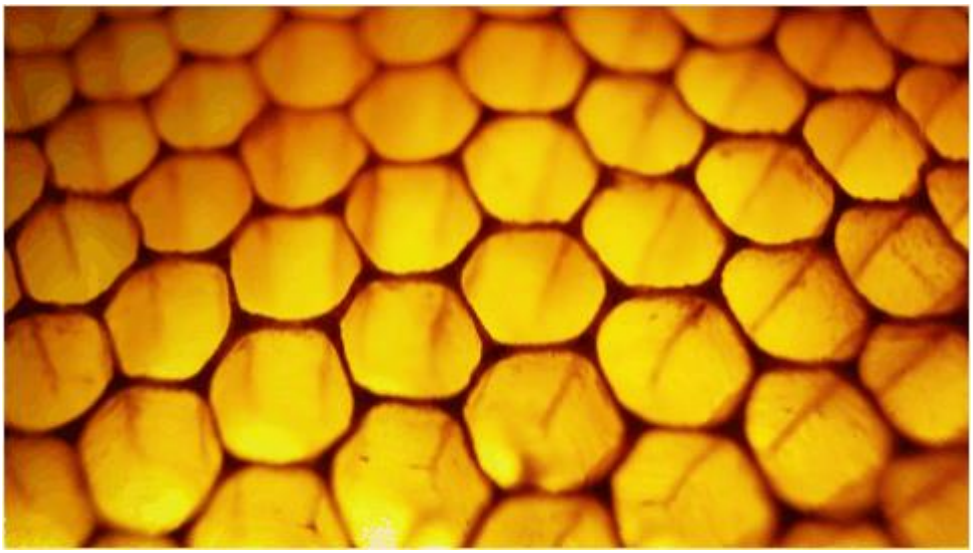


設計の不要な部分を削り、設計の重要点を維持します。 ユーザーがコンテンツに没入できるように、最も重要な要素だけを画面上に残して、簡潔で目的性の高いエクスペリエンスを実現します。

- さまざまな機能で平均点を目指すのではなく、1 つの機能に突出した設計を目指します。
- クロムよりもコンテンツを優先します。
- グラフィックによってユーザーを導き、目的のコンテンツに没入できるようにすることで、ユーザーは自分で探すようになります。
- ユーザーに安心感を与えます。
- UI の冗長さを減らします。

注: Bauhouse デザイン学校が不用なデザイン要素を取り除くように指導したように、弊社でも革や生地の効果、縫い目、影、反射など、現実世界に対するデジタル的な比喩を残すデザインを推奨します。

全体で勝つ

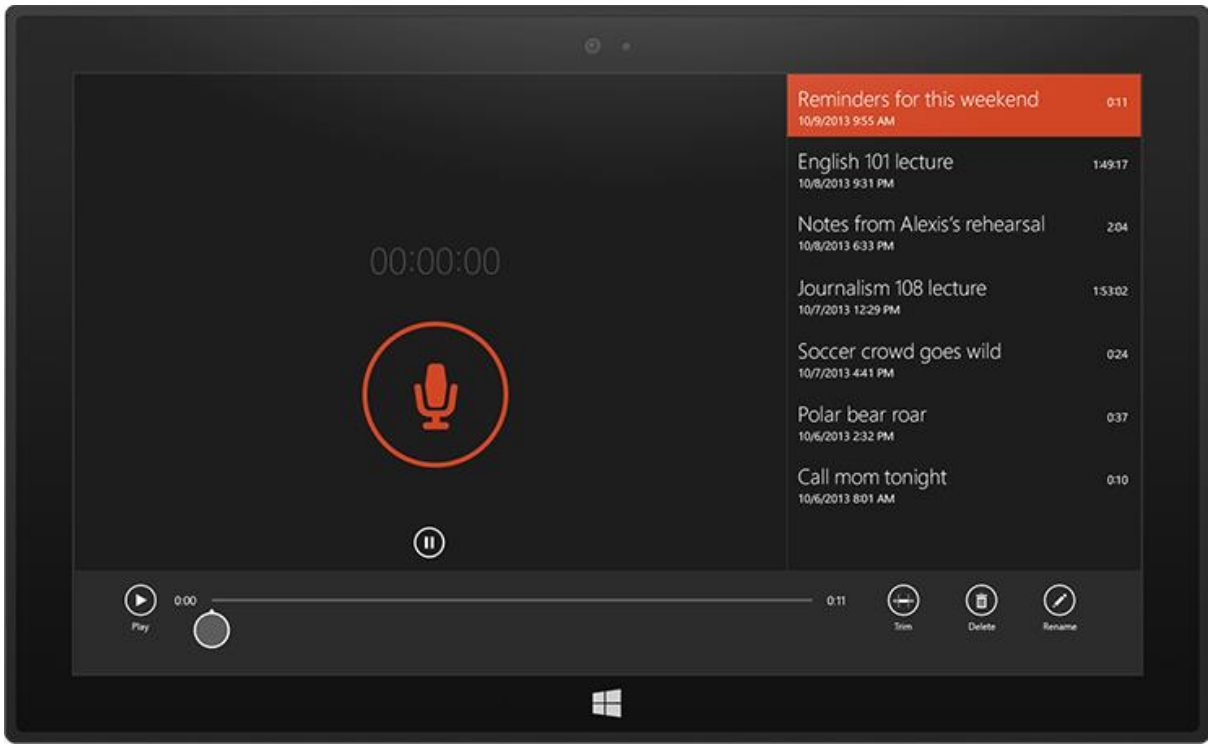


他のアプリ、デバイス、システムと連携してユーザーのシナリオを実現します。たとえば、ユーザーがあるアプリのコンテンツを別のアプリと共有できるようにします。標準的なタッチ ジェスチャやチャームなど、ユーザーが既に知っていることを活かして、親近感、コントロール感、安心感を与えます。

- UI モデルを使います。
- アプリ コントラクトに参加することで他のアプリを操作し、シナリオを実現します。
- マイクロソフトが提供するツールやテンプレートを使って、一貫性を高めます。

Microsoft の設計に関するこの 5 つの原則に従うことで、アプリを設計するときに最適な選択をすることができます。

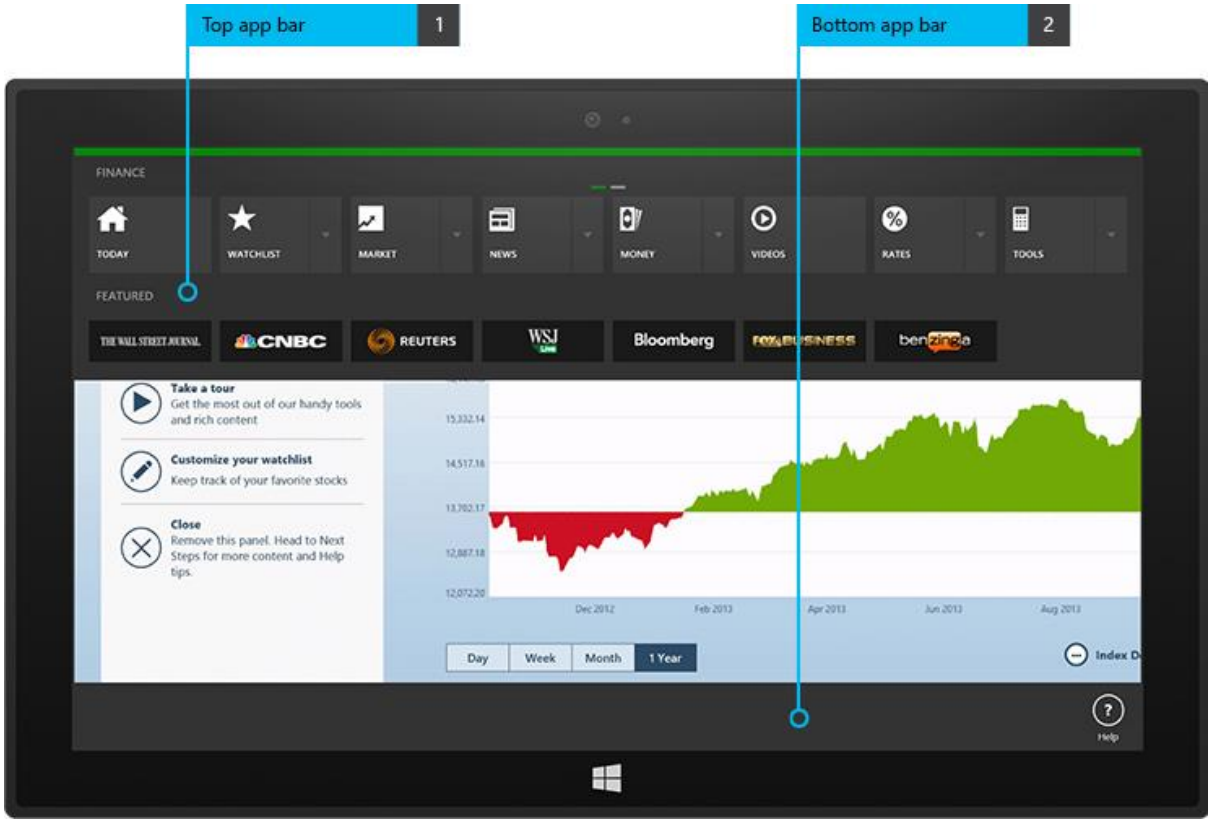
エッジ



エッジは、Windows に統合された強力で美しい対話操作ツールです。エッジのデザインは、"より少ない要素でより大きな効果を上げる" という原則に基づきます。エッジは、より少ない要素でより大きな効果を上げるために一連のオプションが用意されています。現在表示されているコンテンツに関連しないのなら、キャンバスから要素を移動しましょう。デバイスを意識させない信頼性が高く直感的なナビゲーションがユーザーに提供されます。

Windows のエッジ

アプリバー

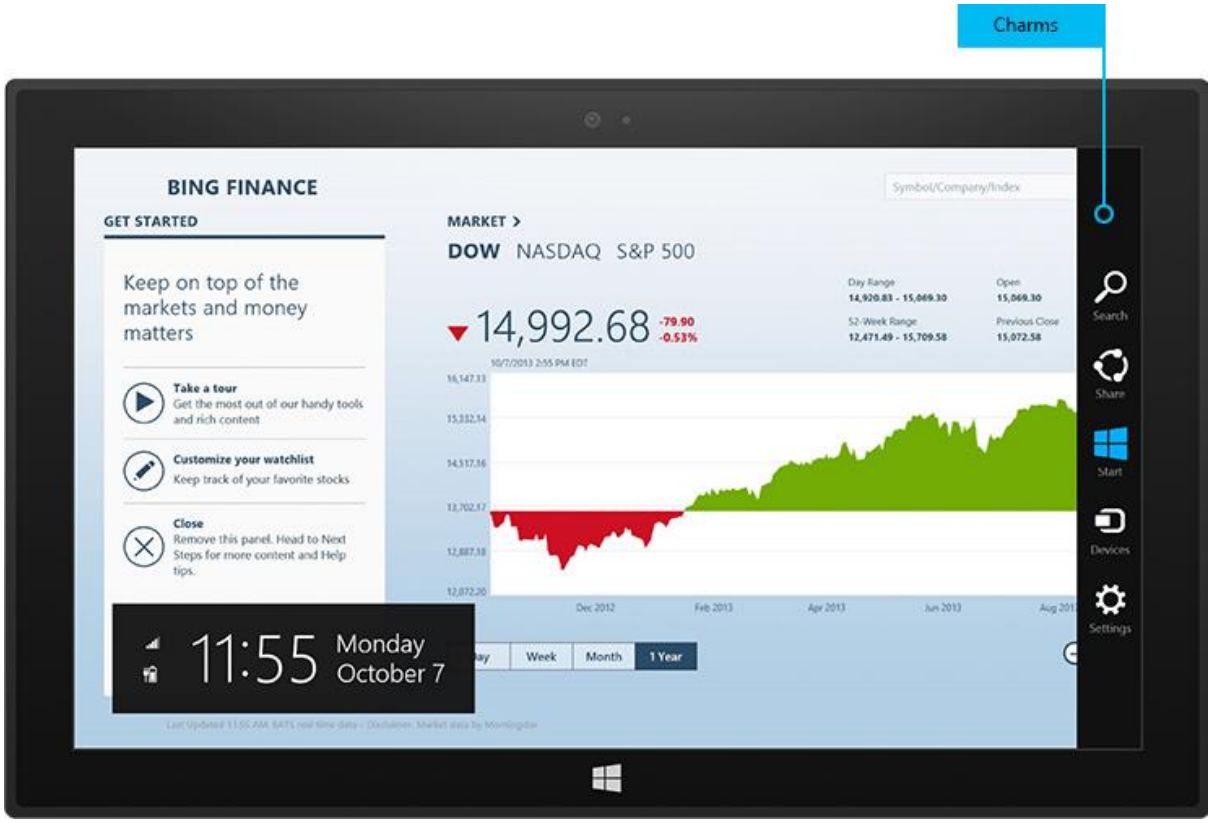


アプリは、上部アプリ バーにナビゲーション コントロールが配置されるのが普通です。この理由から、上部アプリ バーはナビゲーション バーとも呼ばれます。

アプリは、下部アプリ バーにコマンド コントロールが配置されるのが普通です。この理由から、下部アプリ バーはコマンド バーとも呼ばれます。

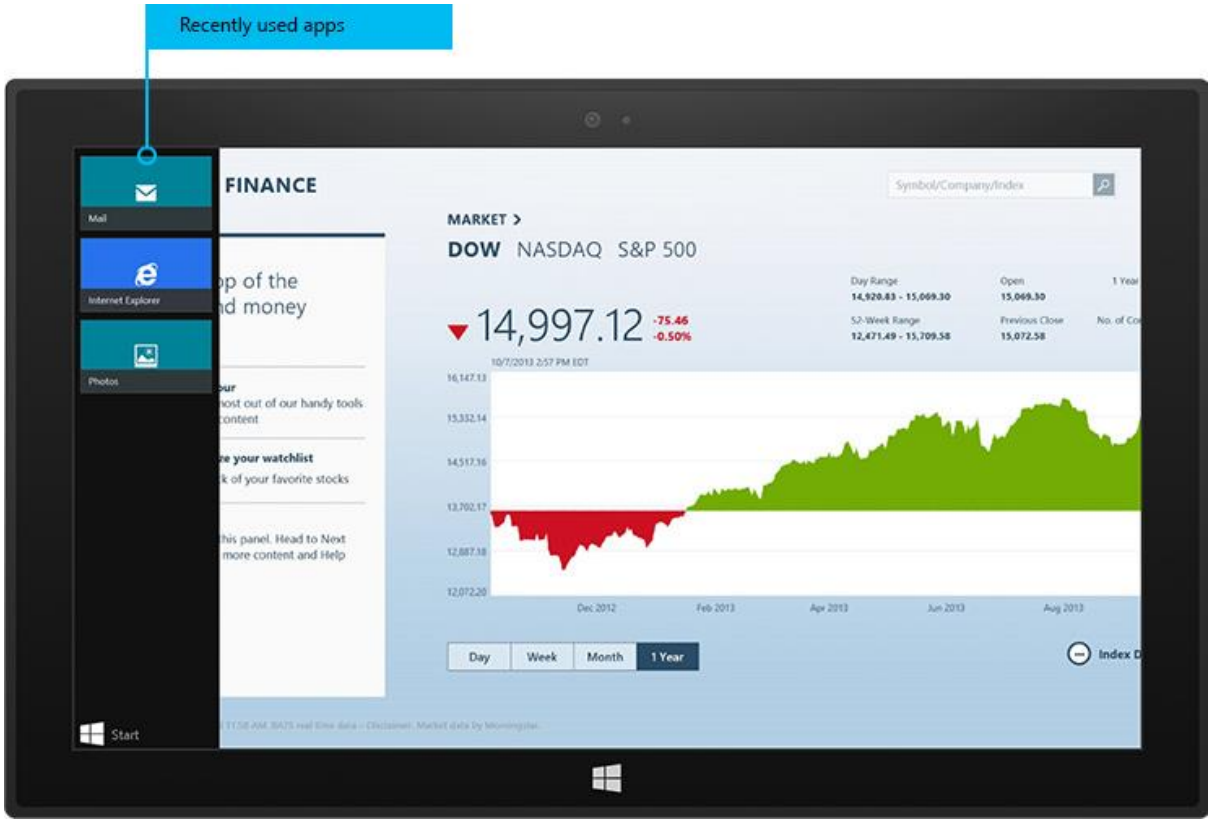
アプリ バーのデザインは、それぞれが行います。システムを操作することはありません。ナビゲーション コントロールとコマンドに加えて、またはその代わりに、その他のコンテキストに応じた関連情報を上部アプリ バーまたは下部アプリ バーに配置できます。たとえば、ショッピング アプリでは、上部アプリ バーにユーザーのショッピング カートのコンテンツを配置できます。この配置により、ユーザーがショッピングに集中しながら、カートとチェックアウト手順をすばやく参照できます。音楽アプリでは、再生、一時停止、早送りなどのメディア コントロールを下部アプリ バーに配置できます。これらのコントロールは、ユーザーが曲を選択するまで非表示であり、選択後にコントロールはコンテキストに関連するものになります。

チャーム



チャームは、エッジの最も象徴的な用途です。チャームを使うことで、システムに対する対話操作を開始します。ユーザーは、画面の右側のエッジからスワイプするか、右上隅または右下隅にポイントすることにより、チャームを起動します。

エッジ（左）



左側のエッジは、アプリの切り替え、最近使用したアプリの表示、画面上で左右に並べた複数のアプリの配置を行うためのユーザーのコマンド センターです。

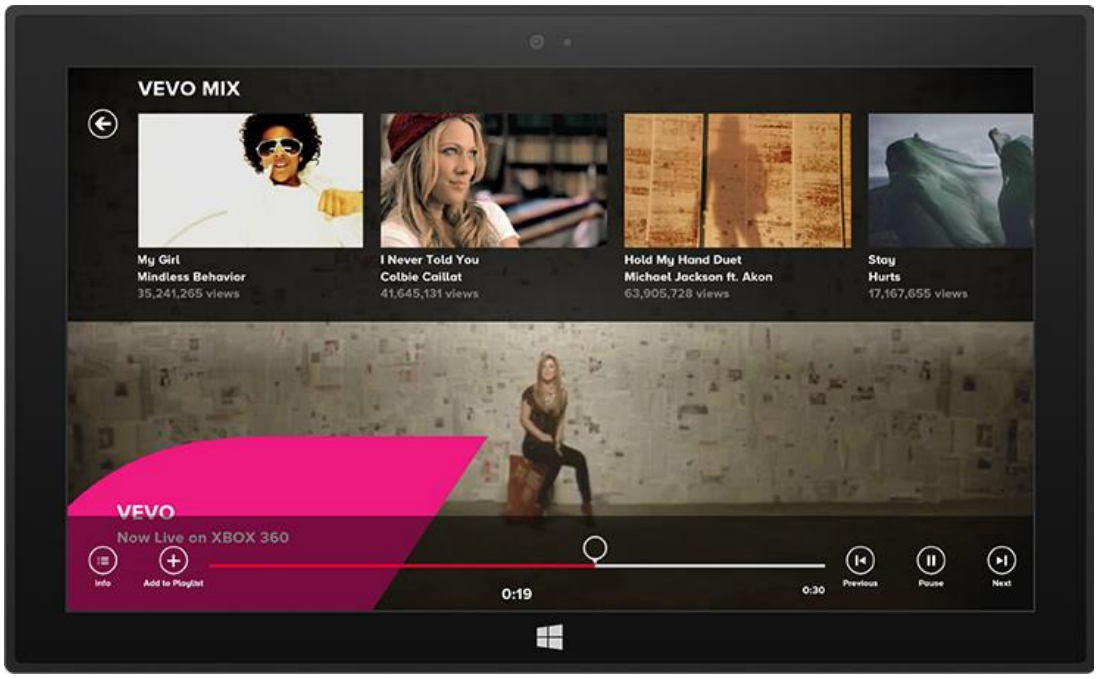
エッジによる技術革新

アプリ デザインでエッジを使うために、新しい革新的な方法を模索することをお勧めします。アプリ バーを実装しないか、最小限の使用に抑えることを選択することもできます。たとえば、Xbox Music アプリでは、キャンバスに大半のコマンドが含まれ、アプリ バーにはコントロールをほとんど配置しません。

標準のコントロールと配置は、単なる出発点でしかありません。アプリでエッジによりどのようなことができるか、創造的に思考してください。

エッジのサンプル

次に例を示します。最初のアプリでは、関連するビデオを表示するために上部アプリ バーを使い、現在再生しているビデオに関連するコマンドのために下部アプリ バーを使っています



NatureSpace アプリでは、下部アプリ バーは非常に小さく、キャンバスの幅全体に広がりませんが、ユーザーが現在のトラックを制御するために使用できるコマンド サーフェスを提供しています。

アプリ バーは、標準テンプレートのような外観にする必要はありません。ブランドに合わせて、美しいデザインにカスタマイズできます。ディズニーのこの例では、アプリ バーは絵本のように見えます。

ライブ タイル

ライブ タイルは、Windows のスタート画面と最新のエクスペリエンスの重要な部分です。ライブ タイルでは、最新のコンテンツと通知が提示されます。ライブ タイルは、軽快で柔

軟な Microsoft デザインの原則を具体化しています。ユーザーはアプリを開く前であっても、スタート画面からすべてのアプリの更新とリアルタイム情報を表示し、新しい情報を迅速に確認してアプリを管理できます。

Windows のライブ タイル

タイルは、スタート画面上のアプリの表現です。タイルには、静的なタイルとライブ タイルがあります。静的なタイルには、既定のコンテンツ (一般に、タイル全体を占めるロゴ画像) が表示されます。一方、ライブ タイルには、ユーザーの関心をアプリに再び集める、新しい価値のあるコンテンツを表示できます。ユーザーの関心を引き付けるために、シンプルなバッジやライブ タイル通知を実装できます。

タイル通知を使って頻繁に情報を更新することで、ユーザーは、アプリが接続され、最新の有益なコンテンツを受け取っていると感じます。タイル通知の更新間隔は、特定のアプリのシナリオによって異なります。たとえば、活発に利用されるソーシャル メディア アプリの場合は 15 分おきに更新し、天気予報アプリは 2 時間おき、ニュース アプリは 1 日に数回、毎日情報を提供するアプリは 1 日に 1 回、雑誌アプリは月に 1 回更新します。アプリでの更新が週に 1 回未満の場合は、古いコンテンツが表示されないように、シンプルなタイルとバッジを併用することを検討します。

通知バッジは、ショート メッセージ サービス アプリなど、短い要約通知をサポートするアプリに最適です。バッジの数値を使って、状態や、ユーザーがアプリを前回起動した時点以降に受信したテキストの数を示すことができます。バッジに常に 50 以上の数値を表示する

可能性が高い場合は、数値の代わりに状態を通知するシステム グリフの使用を検討します。

タイル通知は、ユーザーに関連していて、興味を引く有益な通知である必要があります。通知とは、詳細を表示したりアクションを実行したりするためにアプリを起動するようユーザーを促すものです。

ユーザーはさまざまなサイズにタイルのサイズを変更できます。さまざまなサイズのライブ タイル通知スタイルを調整できます。たとえば、大きなタイルではイメージが非常に魅力的なものになる可能性がありますが、普通サイズのタイルでは簡単な通知をより効率的になることが考えられます。各タイル サイズを考慮し、それに応じてライブ タイルをデザインしてください。

ライブ タイルによる技術革新

ライブ タイルに表示されるコンテンツは、簡単な通知からイメージ、そしてアプリを開いて詳細を確認するようにユーザーに促すより長いコンテンツの抜粋に至るまで、大きく異なります。弊社では、ライブ タイルが提供するコンテンツの可視性と適時性を利用するようにデザイナーと開発者にお勧めしています。

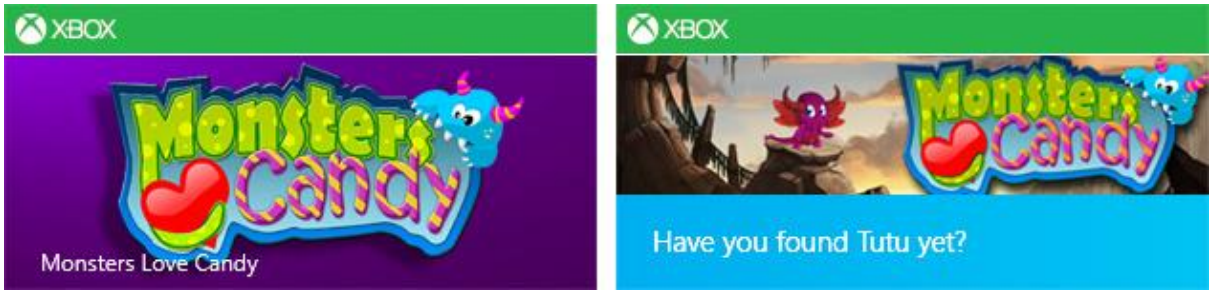
次のような特性がライブ タイルの魅力を高めます。

- 頻繁に更新されるコンテンツ。アプリが実行されていなくても、ユーザーはアプリがアクティブだと感じます。
- ユーザーに関する情報 (アプリの設定を通じてユーザーが指定した興味の対象など) に基づく、カスタマイズされた更新。
- ユーザーの現在の状況に合ったコンテンツ。

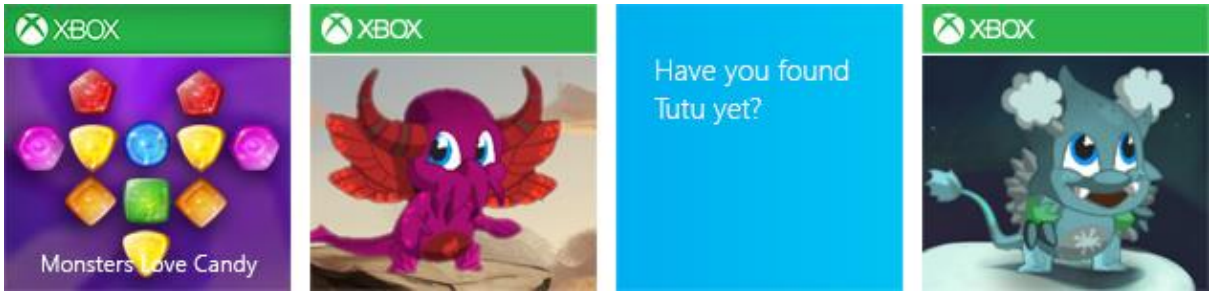
例

Monsters Love Candy

Monsters Love Candy ゲームは、そのブランドを示すためにタイル全体を使っています。ワイド タイルでは、XBOX バナー、ゲームの鮮やかな画像が表示され、より詳細な内部コンテンツを確認するように促されます。



より小さなサイズでは、Monsters Love Candy ライブ タイルがわかりやすく動的です。まず、キャラクター アートを表示し、さらに確認する必要がある情報があることをプレーヤーに通知するテキストにスクロールします。



News Bento

ユーザーに合わせたニュース アプリである News Bento は、さまざまなライブ タイル サイズでニュース記事を伝えます。News Bento は、ユーザーが関心を持つソースからのコンテンツを集めることで、各記事のソースをライブ タイルに示し、アプリに対するユーザーの関心と信頼を高めます。

このアプリは、大きいタイル サイズまたはワイド タイル サイズで、ニュース記事の画像を使います。小さなタイル サイズでは、記事のソースと見出しがタイル全体にわたります。象徴的な News Bento の青い背景によりユーザーは、多様なコンテンツを表示していても、他のタイルの中からこのアプリを簡単に確認できます。

スプラッシュ スクリーン

スプラッシュ スクリーンは、ユーザーが Windows ストア アプリを起動したときに最初に目にするものです。スプラッシュ スクリーンは、ユーザーが Windows ストア アプリを起動したときに最初に目にするものです。スプラッシュ スクリーンにより、リソースを初期化しているときであってもアプリの起動中であることが即座にユーザーに示されます。スプラッシュ スクリーンは、アプリが操作できる状態になるとすぐに、Windows によって閉じられます。ユーザーの没入性を高めるように設計されたスムーズで洗練された読み込みエクスペリエンスを提供します。

Windows のスプラッシュ スクリーン

スプラッシュ スクリーンは、画像と背景色から構成されます。ユーザーには、アプリの読み込み中にアプリのプレビューが表示されます。明確にアプリを識別する画像と配色を使って、さまざまなフォーム ファクターで適切に動作するスプラッシュ スクリーンをデザインします。スプラッシュ スクリーンが表示されるとき、背景のサイズだけがさまざまな画面サイズに合わせて変更されます。透過 .png と単色の背景を使うことによりシンプルさを維

持することもできますし、リッチな写真とアニメーションを使ってユーザーにより適合したエクスペリエンスを作成することも可能です。

Windows ストアは、スプラッシュ スクリーンのデザインについて最小限のアプローチを採用しています。このことは、アプリのスプラッシュ スクリーンで同じ処理を行う必要があることを意味するわけではありません。ユーザーの没入性を高め、ユーザーが引き続きアプリを使用し続けるような強力な第一印象を与える方法を考えます。

スプラッシュ スクリーンによる技術革新

スプラッシュ スクリーンは、単なる色の背景とアイコンに留まるものではありません。 アプリを差別化し、ユーザーに対して優れた第一印象を与えるために、背景とアイコンを共にカスタマイズできます。スプラッシュ スクリーンは、アプリのコンテンツのプレビューであり、アプリのパーソナリティに直接関連させる必要があります。ユーザーの没入性を高め、アプリのエクスペリエンスにユーザーを引き込むために、このブランドの印象付けの機会をうまく使ってください。

例

Phototastic

スプラッシュ スクリーンでは、簡単で単純なアプローチであっても強力な印象を与えることができます。この例では Phototastic のロゴは、かすかな灰色の背景に中央揃えで配置され、このブランドの大胆な色使いを目立たせることができます。

Foursquare

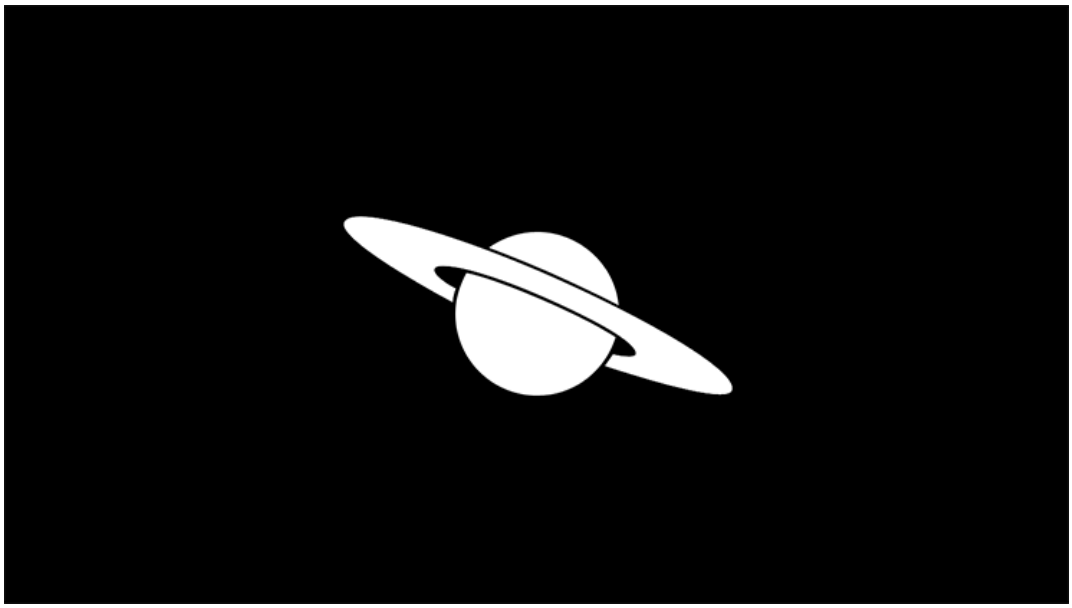
Foursquare アプリが場所の起動と特定を行っている間、スプラッシュ スクリーンでは近隣にある場所の画像を表示し始めます。このリッチでイマーシブな読み込みエクスペリエンスは、新しい場所の発見エンジンとしてのアプリの確約に対して忠実さを維持しています。

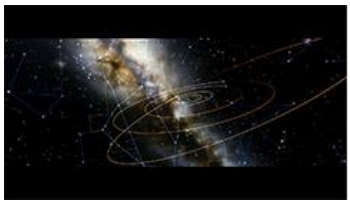
Halo Spartan Assault

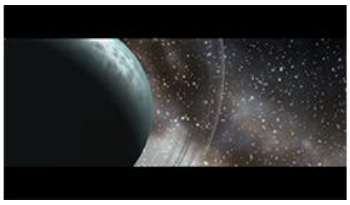
Windows ストアに追加された非常に人気の高い Halo シリーズの 1 つである Halo Spartan Assault では、スプラッシュ スクリーンを映画的なエクスペリエンスとして扱っています。バックグラウンドでゲームを読み込む間、美しいカット シーンにより Spartan Assault がそのコンテキストとして Halo 世界にどのように適合するかを示しています。

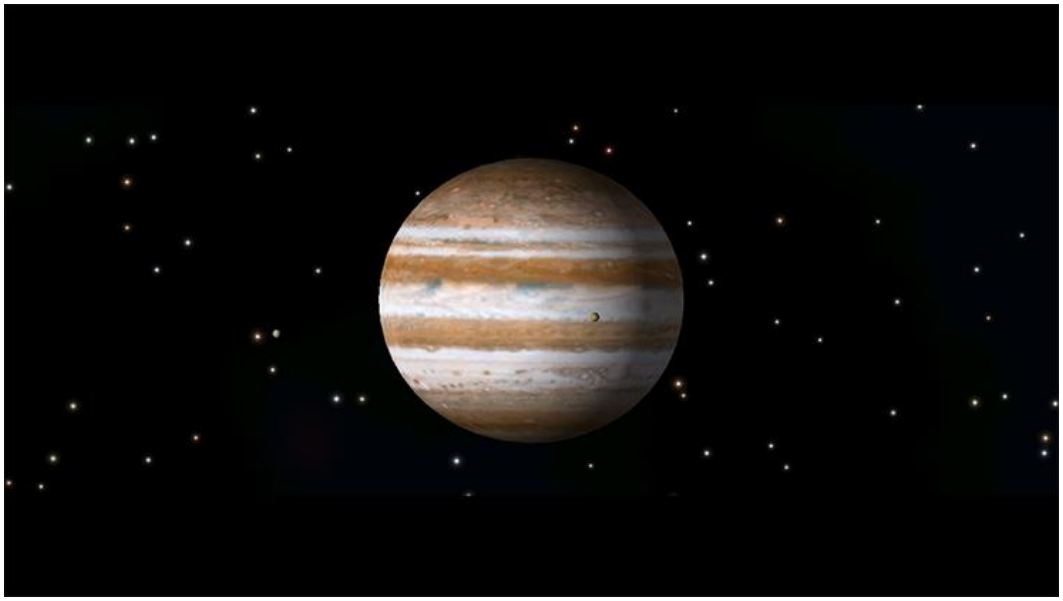
星座表

星座表アプリは、最初のエクスペリエンスをユニークで魅力的なものにするために、複雑なアニメーション シーケンスを使います。スプラッシュ スクリーンのアプリ ロゴを表示した後、太陽系の感動的な表示に切り替えます。これは、ユーザーに提示されるアプリのコンテンツの概要を示す最適なプレビューです。美しい写真は星を眺めるのに適したムードを高め、ユーザーをアプリに招待します。

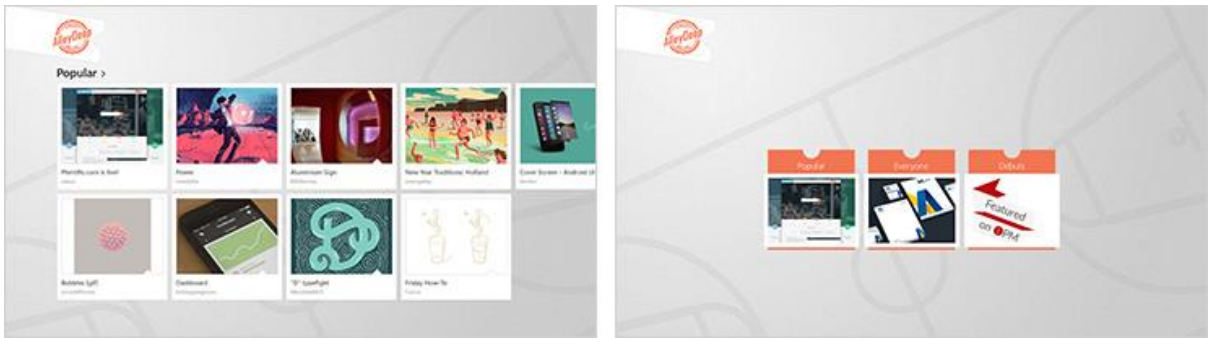






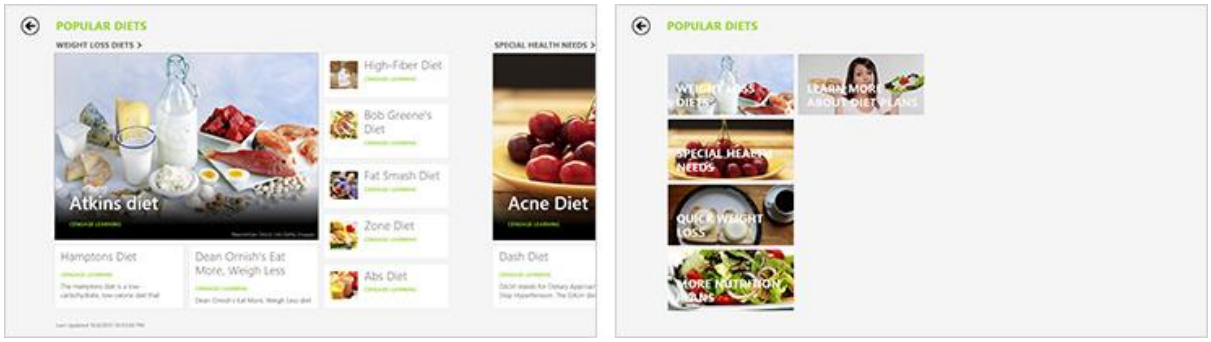


セマンティック ズーム



セマンティック ズームは、強力なナビゲーション ツールです。アプリのコンテンツの概要を示す表示を縮小できます。簡単な光学式ズームの代わりにセマンティック ズームを使うことで、アルファベット順、カテゴリ別、年代順、またはその他の条件でデータをグループ化できます。セマンティック ズームは、軽快で柔軟な Microsoft デザインの原則に基づいて、効率的なナビゲーション手段を提供しながら、コンテンツの概要を示すビューを提供します。また、アイテムの数、プレイしたゲームの数、結果、スコアなど、各グループに関する追加情報を表示することもできます。パンとスクロールだけでもコンテンツのナビゲーションはできますが、セマンティック ズームを使うと、追加のナビゲーションや整理が可能になります。

Windows のセマンティック ズーム



ユーザーは、ピンチ ジェスチャとストレッチ ジェスチャを使ってセマンティック ズームを実行します。コンテンツ領域上で指を広げると拡大表示になり、指を近づけると縮小表示になります。また、Windows ストア アプリで Ctrl キーを押しながら、マウスまたはホイールでスクロールさせることもできます。セマンティック ズームでは、コンテンツを表示するために有用なさまざまな方法を提供することにより、真にデジタルな Microsoft デザインの

原則を利用しています。アプリ内のデータの分類について、ユーザーに迅速な提示を行うことによって、複雑さが軽減し、特定の領域へのナビゲーションが容易になります。

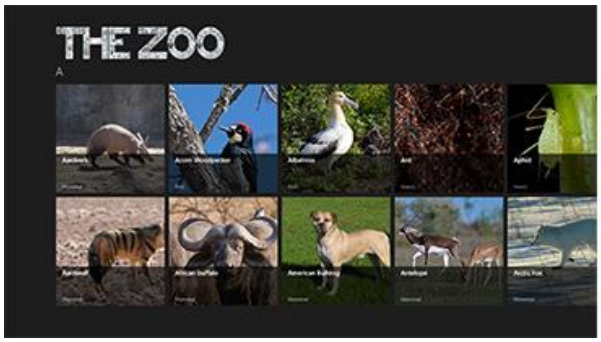
セマンティック ズームによる技術革新

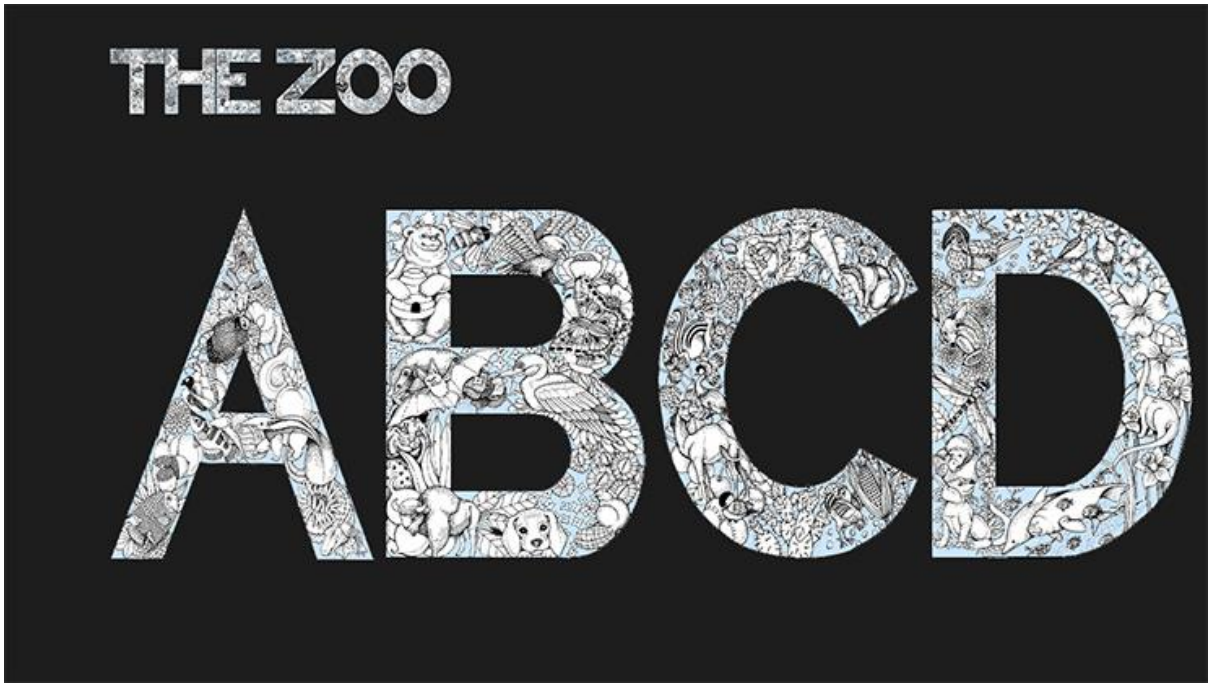
セマンティック ズームによって、ズーム レベルに基づいて、情報の構造と表示を変更できます。最も一般的なユーザー シナリオに対して最適化された魅力的なソリューションを作成できます。セマンティック ズームを利用して、複雑さを軽減し、アプリのコンテンツの組織をよりわかりやすくします。

例

My Zoo

My Zoo アプリは、セマンティック ズームの創造的で楽しい実装の例です。縮小を行うと、コンテンツがアルファベット順に示され、アプリのパーソナリティとブランドがアルファベットの個別の文字に至るまで適用されていることがわかります。各アルファベット文字には、その文字から始まる動物の図が入力されます。この視覚的処理によって、思いもよらない喜びの瞬間を提供しながら、スムーズで容易なナビゲーションが可能になります。



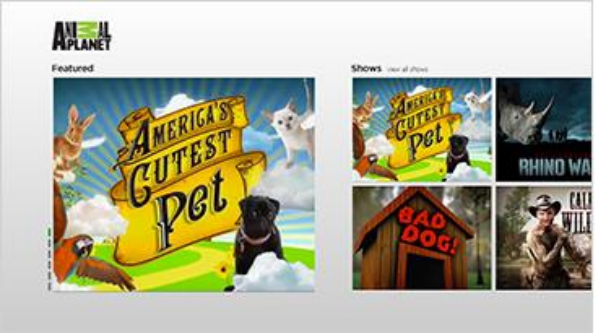


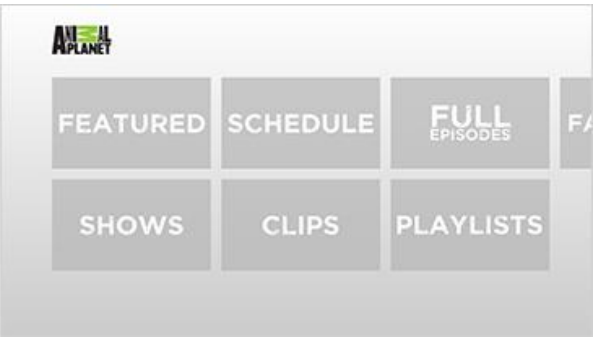
Zappos

Zappos アプリは、顧客に対して迅速で容易なショッピングを実現するためにセマンティック ズームを使っています。従来の拡大表示ではフィルター オプションにより製品のカテゴリとサブカテゴリが示されるのに対して、縮小表示ビューでは簡単な統合レイアウトが提示されます。両表示間の切り替えはスムーズかつ柔軟であり、Zappos が提供する広範なカタログを容易に確認できます。

Animal Planet

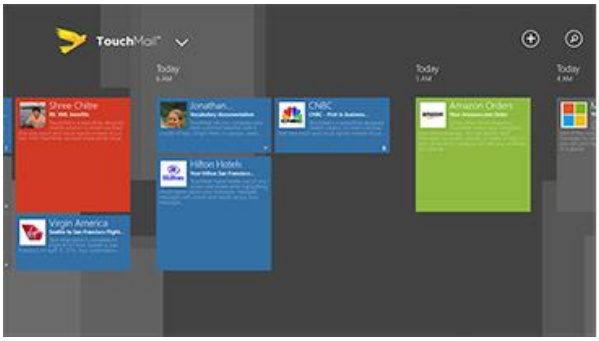
Animal Planet アプリは、セマンティック ズームを使って、明確な文字体裁処理によりカテゴリを単純化することにより、複雑さを軽減しています。拡大表示では、フルカラーの写真、図、回転するコンテンツ カルーセルの利用に適しています。縮小表示では、より落ち着いたグリッドにコンテンツの種類が示され、ユーザーが選択したカテゴリに安心して移動できます。



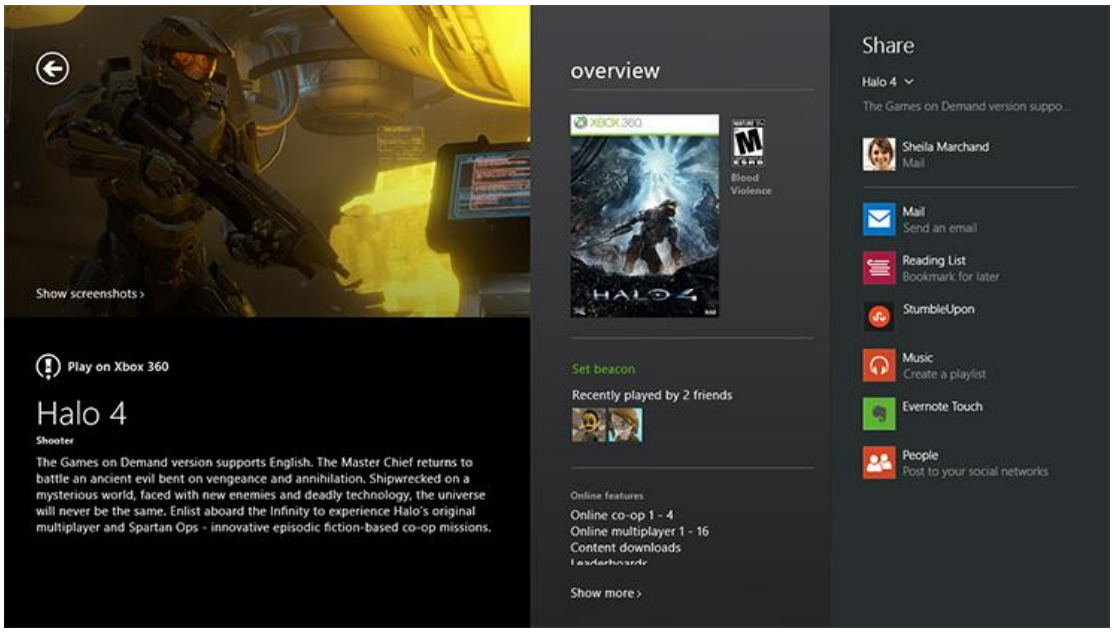


TouchMail

TouchMail は、セマンティック ズームに対する真に革新的なアプローチを採ります。このアプリでは、ズーム レベルを変更すると、受信トレイのコンテンツのリッチな視覚エフェクトで表示される詳細レベルも変更されます。TouchMail は、受信トレイを迅速かつきれいに整理するためのユーザー機能を最適化し、必要時にすぐに重要な電子メールを利用できるようにします。



チャームとコントラクト



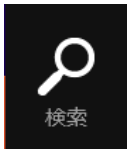
チャームとコントラクトは、コントラクトを通じて他の Windows ストア アプリやシステム UI に結び付けられます。チャームとコントラクトは共に、検索や共有などの最も一般的なタスクを実行する一貫性のある方法を提供します。チャームとコントラクトにより、さまざまなアプリが予測可能なよく知られている方法で相互に対話できます。

チャームとコントラクトは、設定を検索、共有、調整する統一された場所を提供しながら、さまざまなアプリを統合するために、単一の Microsoft デザインの原則の利点を集約しています。

Windows のチャーム

ユーザーが画面の右側のエッジからスワイプするか、右上隅または右下隅にポイントすることによりアクセスできる 5 つのチャームがあります。またチャームでは、アプリを操作する想定された方法を提供するほか、アプリでのデータの共有を可能にします。異なるアプリ間でイメージを受け渡すこと、アプリから直接外部ソースに情報を電子メールで送信すること、さらに他のアプリから起動する検索クエリに用語を提示することができます。

5 つのチャームを次に示します。

	**検索** このチャームを使うと、現在開いているアプリに加えて、他のアプリ、ファイル、または Web を対象に検索を実行できます。
	**共有** このチャームを使うと、現在のアプリケーションの外部のコンテンツを共有できます。
	**スタート** このチャームは、スタート画面を表示します。
	**デバイス** このチャームを使うと、コンテンツを印刷したり、リモート再生デバイスに送信したりできます。
	**設定** このチャームにより、システムのアプリ設定にアクセスするための、信頼性の高い一貫した方法が提供されます。

Windows 内のコントラクト

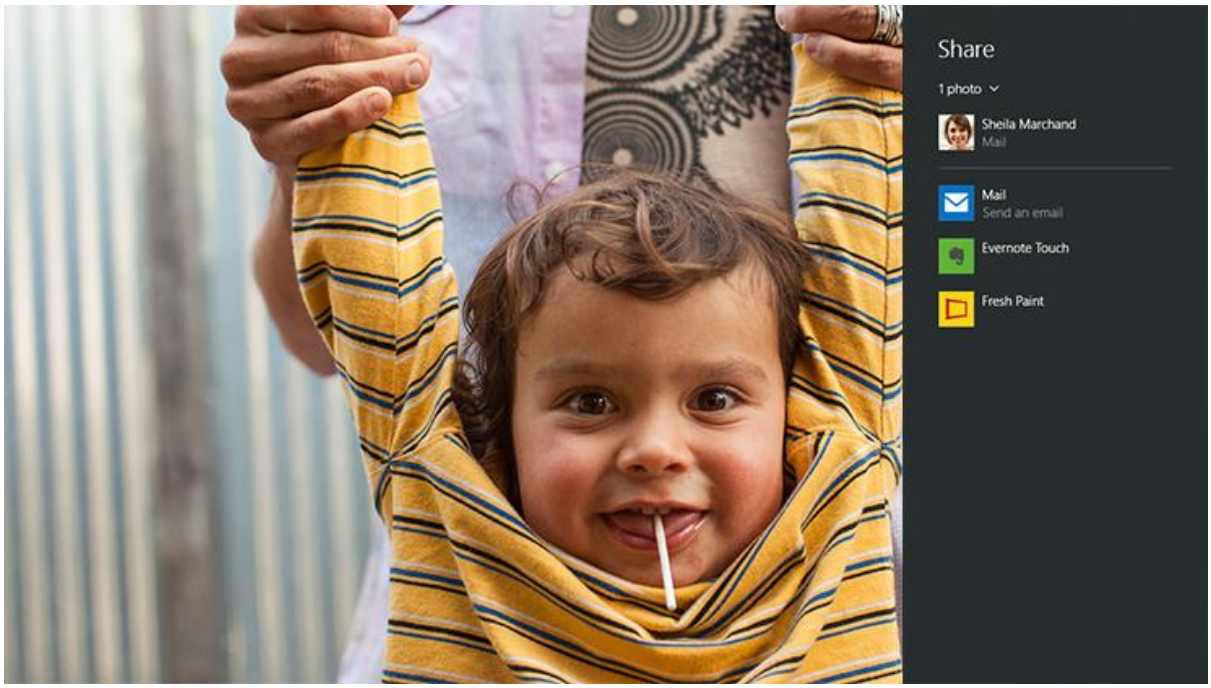
Windows 8.1 では、さまざまなエンド ツー エンドのシナリオに対応するために、チャームはコントラクトと組み合わせて動作します。コントラクトとは、1 つ以上のアプリの間の契約のようなものです。コントラクトは、アプリが独特な Windows の対話的操作セットに参加するために満たす必要がある要件を定義します。 たとえば、Windows ではコントラクトを使用することによってユーザーがあるアプリのコンテンツを別のアプリと共有できます。コンテンツを分け与えるアプリは、特定の要件を満たすことでソース コントラクトをサポ

ートします。一方、共有コンテンツを受け取るアプリは、別の要件セットを満たすことでターゲット コントラクトをサポートします。アプリどうしが認識し合う必要はありません。共有コントラクトに参加するアプリはすべて、Windows によって共有ワークフローがエンド ツー エンドで完全にサポートされることを信頼できます。

Windows には、チャームによって呼び出される機能に加えて、多くのコントラクトが用意されています。たとえば、ローカル PC、接続された記憶装置、ホームグループ、他のアプリからユーザーがファイルを選べるようにすることができます。アプリでは位置認識を提供できます。また、マルチメディアに対応することもできます。

チャームとコントラクトによる技術革新

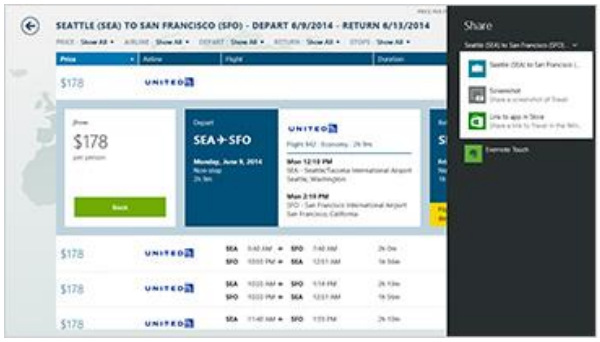
アプリ間でコンテンツを共有する多くのシナリオが考えられます。ユーザーは、メディア コンテンツを検索および共有することや、再生リストを作成し単純に共有チャームを起動して友人に再生リストを送信することができます。またユーザーは PC から、テレビ、オーディオ/ビデオ レシーバーなどのネットワーク接続デバイスにコンテンツを送信し、表示をグループ化できます。[共有] ウィンドウに表示される時間節約のためのリンクである QuickLinks をアプリに追加できます。これは、ユーザーに合わせてカスタマイズされており、よく使われる連絡先に対してコンテンツを迅速に共有できます。

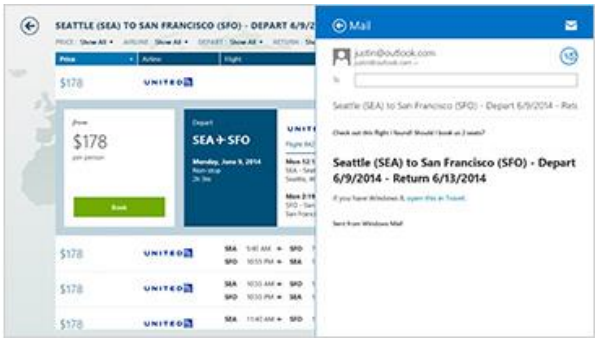


例

Bing トラベル

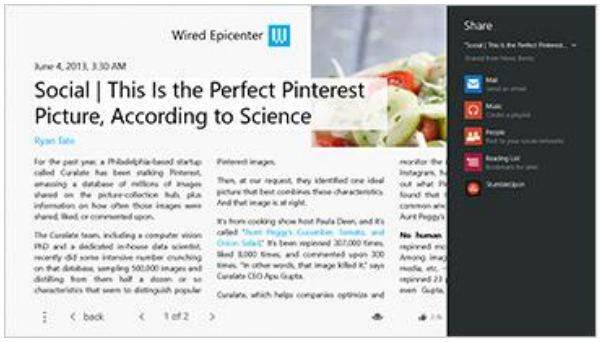
この例では Bing トラベルは、電子メールによって連絡先に全旅行計画を迅速かつ簡単に転送するために、共有チャームを使ってユーザーに代わって効率性を最適化しています。





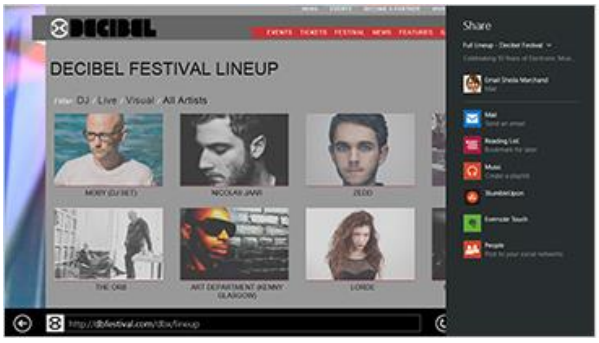
Wired Epicenter

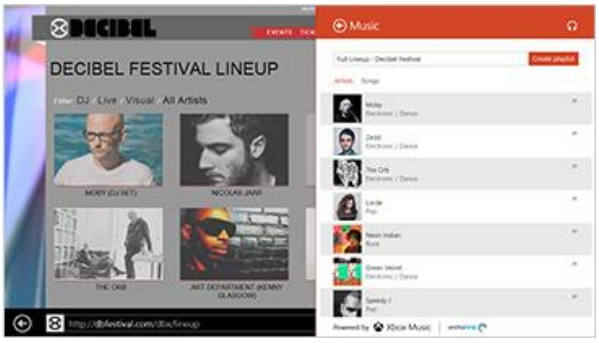
Wired Epicenter アプリで示されているように、あるアプリから別のアプリにコンテンツを共有することにより、コンテキストを保持しながら生産性を最大限に高めることができます。この例では、ユーザーは現在のコンテンツから移動することなく、Facebook に対して記事を共有できます。



Xbox Music

データの共有は、あるアプリから別のアプリへのコンテンツの単純な移動に留まるわけではありません。次の例では、音楽 Web サイト、この場合は Decibel's Festival Lineup からデータを収集し、再生リストにまとめる方法を示しています。ユーザーは個別の曲を検索してそのタイトルをコピーおよび貼り付けする必要はありません。その代わりにユーザーは、共有チャームをタップし、Xbox Music を選択するだけです。再生リストを作成すると、Windows によって残りの作業が行われます。





ビジョン

アプリを計画するには、アプリがユーザーに提供するエクスペリンスにおいて明確なビジョンを定義することが必要です。他の表現にすれば、「アプリの目的は何か」と問いかけて、アプリを利用するユーザーが達成することを明確にします。

ビジョンと計画

期待を上回る結果を得る

成功するビジョンを構築するための手順を理解してください。時間とスキルは貴重なものです。成功に導き、確固とした計画を作成するために投資してください。

レスポンシブ デザインとフォーム ファクター

フォーム ファクターの多様なセットを考慮してください。

複数のフォーム ファクターに向けてアプリのエクスペリエンスをデザインしてください。そして、アプリが流れるように動くために、レスポンシブ デザインを活用してください。

移行

Windows によって製品の質をさらに高めてくださいカスタマイズされたデザイン アプローチと豊富なプラットフォーム ツール セットを使って、Windows にアプリを移行します。

移行例とガイダンスを参照してください。

カテゴリのアイディア

Windows ストア アプリのカテゴリについては、アイディア ブックからヒントを見つけてください。

アイデア ブックをご覧ください。

ビジョンの定義

Windows ストア アプリを計画する際は、アプリに追加する機能のことよりも、どのようなユーザー エクスペリエンスを提供するかについて考えてください。

Windows ストア アプリを計画する際には、次の手順を行うことをお勧めします。

1. アプリの長所を決める

Windows ストア アプリを設計する際に最も重要なのは、アプリの特徴を決定することです。たとえば、写真アプリを作るとします。まず、ユーザーが写真を操作、保存、共有する動機を考えてみます。それは、写真で思い出をたどったり、写真を通じて他の人々と交流したり、写真を安全に保管したりすることであるとわかります。このような用途に適した写真アプリを作ることにします。このユーザー エクスペリエンスの目標を意識しながら、アプリの設計プロセスを進めていきましょう。

何のためのアプリか まず、大まかなコンセプトを決めます。ユーザーがアプリで何をできるようにするか、一覧にまとめてみましょう。

たとえば、旅行の計画に使うアプリを作るとします。次のように、頭に浮かんだアイデアをメモしてみましょう。

- 旅行計画に含まれている場所の地図をすべて集めて、旅行中もそれを携帯する。
- 旅行先で滞在中に開催されるイベントを見つける。
- 必ずするアクティビティや必ず見る観光名所の一覧を、旅行に行くメンバーが各自で作成し、他のメンバーも見られるようにする。
- 旅行のメンバーが撮ったすべての写真をまとめて、友人や家族と共有する。
- 航空料金に基づいて、お勧めの行き先を選ぶ。
- 目的地周辺のレストラン、店舗、アクティビティが多数掲載されている情報源を見つける。

アプリの 1 番の特徴は何か すべてのアイデアを全体的に見て、特に目立つシナリオがないか考えてみます。数多くのアイデアの中から絞り込んで、これに集中しようと思えるシナリオを 1 つだけ選びます。優れたアイデアをたくさん捨てることになりますが、その 1 つのシナリオを良いものにしあげるには、いさぎよくアイデアを捨てることが肝心です。

シナリオを 1 つ選んだら、アプリの 1 番の特徴を普通の人に 1 行で伝えるにはどうしたらよいかを考えます。 例:

- この旅行アプリを使えば、友人どうしで協力してグループ旅行の計画を作り上げることができます。
- このワークアウト アプリを使うと、友人どうしでトレーニングの経過を記録して、互いに成果を見せることができます。
- この食料雑貨アプリを使うと、家族で毎週の食料雑貨を計画的に買うことができ、買い忘れや重複がなくなります。

このようにアプリの 1 番の特徴を示す説明文を作っておくと、アプリを作るプロセスで、設計上のさまざまな決定事項や妥協点を判断する際に役立ちます。アプリの 1 番の特徴にするユーザー シナリオを中心とした説明文にしますが、単なる機能一覧にならないように注意してください。この説明文では、アプリが実行できる機能を説明するのではなく、ユーザーがこのアプリを使ってできることを説明します。

この手順に役立つ一般的な方法: ブレーンストーミング、ダイアグラム、マインド マッピング

2. サポートするユーザーのアクティビティを決める

フローとは、ユーザーが目的を果たすためにアプリで行う一連の操作です。すべてのフローは、"1 番の特徴" の説明文と関連していることが必要です。アプリ用に選んだ 1 つのシナリオを実現できるように、フローを作ります。優れたアプリは、覚えやすくて操作が少ないフローを備えています。

この手順に役立つ一般的な方法:

- **フローの概要**　最初の操作とその次の操作を決める
- **フローのストーリーボード**　フロー完了までの UI 操作の順序を決める
- **プロトタイプ**　簡単なプロトタイプを使ってフローをテストする

ユーザーは何をできるか たとえば、旅行アプリを使うと "友人どうしで協力してグループ旅行の計画を作り上げる" ことができます。必要なフローをリストにまとめましょう。

- 一般的な情報で旅行計画を作る。
- 友人たちを旅行に誘う。
- 友人の旅行に参加する。
- 他の旅行者が勧める旅行計画を見る。
- 目的地とアクティビティを旅行に追加する。
- 友人たちが追加した目的地とアクティビティを編集したり、コメントを書いたりする。
- 友人や家族に見てもらえるように旅行計画を共有する。

3. アプリに含める機能を決める

ユーザーの目的を理解し、その目的を助ける方法もわかったら、次にすることは、それを実現するための機能を探すことです。Windows プラットフォームを調べて、アプリのニーズに対応する機能を探します。各機能については、ユーザー エクスペリエンス (UX) のガイドラインに従ってください。

一般的な方法:

- **プラットフォームの調査**　プラットフォームにある機能を確かめて、どのように使えるかを考える。

- **関連付けのダイアグラム**　フローと機能を結び付ける。
- **プロトタイプ**　機能をテストして、必要な働きができるかを確かめる。

アプリ コントラクト　他のアプリとのユーザー フローや他の機能とのユーザー フローを実現するアプリ コントラクトに、アプリを参加させることができます。

- **検索**　ユーザーがアプリのコンテンツを、システムのあらゆる場所から (別のアプリの内部からも) すばやく検索できるようにします。逆も同じように検索できます。
- **共有**　ユーザーがアプリのコンテンツを他のアプリ経由で他のユーザーと共有できるだけでなく、他のユーザーやアプリの共有コンテンツを受け取ることもできるようにします。
- **リモート再生**　オーディオ、ビデオ、または画像をアプリからホーム ネットワーク上の別のデバイスにストリーミングして楽しむことができます。
- **ファイル ピッカーとファイル ピッカーの拡張機能**　ユーザーがローカル ファイル システム、接続された記憶装置、ホームグループ、さらに他のアプリからファイルをアプリに読み込み、保存できるようにします。また、ファイル ピッカーの拡張機能を使って、アプリのコンテンツを他のアプリに読み込むこともできます。

詳しくは、アプリ コントラクトと拡張機能についてのページをご覧ください。

さまざまなビュー、フォーム ファクター、ハードウェア構成　Windows 8.1 では、ユーザーがさまざまな持ち方や向きでデバイスを構えるので、アプリもそれに対応する必要があります。アプリの UI は、ユーザーが使うあらゆるデバイス、入力モード、向き、ハードウェア構成、状況で適切に表示される必要があります。

タッチ優先　Windows 8.1 では、独自の直感的なタッチ操作を使うことができます。これはマウスと同じように使えるだけではありません。

たとえば、セマンティック ズームは、大きなコンテンツ セットの中を移動するときにタッチ操作のメリットが活かした操作方法です。パンまたはスクロールでコンテンツのカテゴリを移動し、カテゴリを拡大して詳しい情報を見ることができます。タブなどの従来のナビゲーション手法やレイアウト パターンよりも、コンテンツを感覚的、ビジュアルに表現でき、多くの情報を示すことができます。

もちろん、回転、パン、スワイプ、クロススライドなどタッチ操作のさまざまな長所を利用できます。

魅力と新鮮さ アプリが新鮮に感じられ、次のような標準的なエクスペリエンスでユーザーを魅了できることが重要です。

- **アニメーション** アニメーションのライブラリを使って、アプリに軽快で柔軟な印象を加えることができます。コンテキストの変化がわかりやすく、ビジュアルの切り替えがエクスペリエンスに結び付きます。
- **トースト通知** トースト通知によって、即時性が必要なコンテンツや個人的に関係のあるコンテンツをユーザーに知らせ、アプリが非表示の場合は、アプリを表示するようにユーザーにうながします。
- **セカンダリ タイル** スタート画面で、アプリの興味深いコンテンツやディープ リンクを紹介します。ユーザーは、アプリ内の特定のページまたはビューを直接起動できます。
- **アプリ タイル** ユーザーと関連性の深い最新の更新情報を知らせて、ユーザーがアプリを開くように誘います。

個人設定

- **設定** アプリ設定を保存することによって、ユーザーは好みのエクスペリエンスを作ることができます。すべての設定を 1 つの場所にまとめて、ユーザーが使い慣れた一般的なメカニズムでアプリを構成できるようにします。
- **ローミング** データのローミングによってデバイス間で連続性のあるエクスペリエンスを実現し、ユーザーが作業を中断したところから再開できるようにすると共に、使われているデバイスに関係なく、ユーザーが最も重視する UX を保持できるようにします。設定と状態をローミングによって維持することで、キッチンで家族が使う PC から仕事用 PC、個人用のタブレットまで、どこにいてもアプリを快適に使えるようにします。
- **ユーザー タイル** ユーザーがアプリを個人設定で使えるようにします。ユーザーの画像を読み込んだり、アプリからのコンテンツを Windows で個人用タイルとして設定できるようにします。

デバイスの機能 最新のデバイスの機能をアプリで十分に活用してください。

- **近接ジェスチャー** ユーザーは、複数のデバイスを物理的に "タップ" することで接続して、複数のユーザーが物理的に近くにいる必要がある状況 (マルチプレイヤーゲーム) での使い勝手を良くすることができます。
- **カメラと外部記憶装置** 内蔵カメラまたは接続されたカメラを利用します。チャットや電話会議、ビデオ ログの記録、プロファイル写真の撮影、日常的な文書作成など、アプリの特徴に応じたさまざまな活動に使うことができます。
- **加速度計やその他のセンサー** 最新のデバイスはさまざまなセンサーを備えています。アプリでは、環境光に応じてディスプレイの明るさを調節したり、ユーザーがディスプレイの向きを変えたときに UI を自動的に再配置したり、物理的な動きに応じて処理を行ったりできます。
- **地理位置情報** 標準的な Web データまたは地理位置情報センサーからの地理位置情報を使って、ユーザーの移動や地図上での位置確認、または近くのユーザー、アクティビティ、目的地に関するユーザー通知を支援します。

旅行アプリの例についてもう一度考えてみましょう。友人どうしで協力してグループ旅行の計画を作り上げるアプリには、次のような機能があるとよいかもしれません。

- **共有:** 次の旅行の計画を複数のソーシャル ネットワークで公開して、旅行前の楽しみを家族や友人と分かち合います。
- **検索** 他の共有された旅行計画や公開された旅行計画を検索して、アクティビティや目的地を探し、自分たちの旅行に取り入れることができます。
- **通知** 旅行の仲間が旅行計画を更新したときに、通知を受け取ります。
- **設定** 通知する旅行、旅行計画の検索用を許可するソーシャル グループなどの設定をユーザーが構成できます。
- **セマンティック ズーム** ユーザーは旅行計画のタイムラインを閲覧しながら、数多くの計画済みアクティビティを必要に応じて拡大表示し、詳しい内容を見ることができます。
- **ユーザー タイル** 友人と旅行を共有するときに表示する画像を選択できます。

このように、Windows 8.1 を使って、ユーザーを喜ばせる魅力的なエクスペリエンスを作ることができます。

4. アプリで収益を得る方法を決定する

さまざまな方法でアプリから収益を得られます。アプリ内の広告や販売を使う場合は、それに対応するように UI を設計します。

5. アプリの UX を設計する

ここでは、適切な基本構造を作ります。アプリの 1 番の特徴が決まり、サポートするフローも決まったので、次はユーザー エクスペリエンス (UX) 設計の基本構造を検討します。

UI コンテンツをどのようにまとめるか ほとんどのアプリ コンテンツは、特定の形式のグループまたは階層にまとめることができます。コンテンツの最上位グループの内容は、1 番の特徴の説明文に沿ったものにします。

旅行アプリを例に考えてみましょう。旅行計画をグループ化する方法はいくつかあります。アプリの主な目的が興味深い目的地を見つけることであれば、冒険旅行、リゾート旅行、ロマンチックな休暇旅行など、興味の対象ごとにグループ化できます。しかし、このアプリの主な目的は友人と協力して旅行計画を立てることなので、家族、友人、同僚など、ソーシャル グループごとに旅行計画を整理する方が理にかなっています。

コンテンツのグループ化の方法が決まると、アプリにどのようなページやビューが必要なのかがわかります。Microsoft Visual Studio に備えられているプロジェクト テンプレートでは一般的なアプリ レイアウト パターンが用意されており、ほとんどのコンテンツに対応できます。

UI コンテンツをどのように表示するか UI をまとめる方法が決まったら、UI がどのように構築され、ユーザーに表示されるかを指定する、UX の目標を定義できます。どのシナリオでも、ユーザーがアプリを楽しんで使い続けることができるよう、できるだけ早くユーザーの使いたくなるという気持ちを高める必要があります。これを行うには、最初にユーザーに表示する UI の部分を決めて、その部分を完成させてから、その他の重要でない部分の構築に時間を使う必要があります。

旅行アプリを例として見てみましょう。ユーザーはアプリを開始して、まず具体的な旅行計画を探します。この情報をできるだけ早くユーザーに表示するには、まず高速で応答性の高

い ListView コントロールを活用して、旅行の一覧を表示します。旅行の一覧が表示されると、友人の旅行のニュース フィードなどの機能の読み込みが始まります。

必要な UI サーフェスとコマンドは何か 前の手順で決めたフローをもう一度見てみます。各フローについて、ユーザーが行う手順を大まかに作成します。

たとえば、"友人や家族が旅行計画を確認できるように共有する" フローで考えてみましょう。ユーザーが既に旅行計画を作成しているとします。旅行計画を共有するには、次の手順が必要です。

1. アプリを開き、作成した旅行の一覧を表示します。
2. 共有する旅行をタップします。
3. 旅行の詳しい内容が画面に表示されます。
4. 共有を開始するための UI を操作します。
5. 旅行を共有したい友人のメール アドレスまたは名前を選ぶか、入力します。
6. 共有の操作を完了するための UI を操作します。
7. アプリの旅行の情報が更新され、旅行計画を共有するメンバーの一覧が反映されます。

このプロセスによって、作成する必要のある UI と、さらに作り込みが必要な部分 (アプリをまだ使っていない友人に送るメールの定型文を考えるなど) を把握することができます。さらに、不要な手順を削る作業も始めます。たとえば、旅行の詳しい内容を共有前に実際に見る必要はないかもしれません。フローが整理されると使いやすくなります。

さまざまなサーフェスを使う方法について詳しくは、Windows ストア アプリのコマンド実行の設計に関するトピックをご覧ください。

フローをどのようにするか　ユーザーの手順を定義したら、そのフローをパフォーマンスの目標にします。詳しくは、「パフォーマンスの計画」をご覧ください。

コマンドをどのように配置するか 大まかなフロー手順に従って、設計する必要のあるコマンドを決めます。次に、アプリのどこにコマンドを配置するかを考えます。

- **できるだけコンテンツを直接操作できるようにします。** できる限り、コンテンツを操作するコマンドを用意せず、アプリのキャンバス上でユーザーがコンテンツを直接操作

できるようにします。たとえば、旅行アプリで旅行計画を編集するときに、リスト内のアクティビティを上下に移動するコマンド ボタンを使うのではなく、キャンバスのリスト上でアクティビティをドラッグ アンド ドロップできるようにします。

- **コンテンツを直接操作できない場合、**コマンドを次の UI サーフェスのいずれかに配置します。
  - **アプリ バー**　コマンドはできる限りアプリ バーに配置してください。通常、アプリ バーは非表示になっており、ユーザーの操作で表示されます。
  - **アプリのキャンバス上**　目的が 1 つに限られているページまたはビューは、その目的用のコマンドをキャンバス上に直接配置できます。しかし、このようなコマンドはなるべく作らないでください。
  - **コンテキスト メニュー**　クリップボードの操作 (切り取り、コピー、貼り付けなど) や、選択できないコンテンツに実行するコマンド (地図の目的の位置にピンを追加するなど) に使うことができます。

各ビューでアプリを準備する方法を決定します。 Windows 8.1 では横向きと縦向きがサポートされ、全画面表示から最小幅にいたるまで、アプリの幅をあらゆるサイズに変更できます。アプリは、任意のサイズの任意の画面で、どの向きでも適切に表示され、機能することが望まれます。つまり、さまざまなサイズとビューについて、UI 要素のレイアウトを計画する必要があります。そうすることで、ユーザーのニーズと好みに合わせてアプリ UI が柔軟に変化します。

詳しくは、「レイアウトの選択」、「ウィンドウ サイズと画面に合わせたスケーリングのガイドライン」と「幅の狭いレイアウトのガイドライン」をご覧ください。

6. 第一印象を良くする

初めてアプリを起動したユーザーがどのように思ったり感じたりするかを考えましょう。アプリの "1 番の特徴" の説明文を再確認します。アプリの 1 番の特徴をユーザーに直接説明できなくても、第一印象を演出することでメッセージを伝えることができます。次の機能を利用します。

タイルと通知　タイルはアプリの顔です。ユーザーのスタート画面に表示される数多くのアプリの中から自分のアプリが起動される決め手になるのは、何でしょうか。タイルでアプリのブランドを明確にして、アプリの特徴を示すようにします。アプリで常に最新の役立つ情報が伝えられるようにタイル通知を使うことで、ユーザーがアプリを繰り返し利用するようにします。

スプラッシュ スクリーン　スプラッシュ スクリーンはできるだけ短時間で読み込む必要があります。この画面は、アプリの状態を初期化する必要が生じない限り画面上に維持する必要があります。スプラッシュ スクリーンの表示内容で、アプリの特徴を伝えます。

初めての起動　ユーザーがサービスの新規登録、アカウントへのログイン、独自のコンテンツの追加を行う前に、ユーザーに対して何を表示しますか。ユーザーに情報を求める前に、アプリの価値をアピールします。アプリを使ってもらうためには、ユーザーが自由に試せるサンプル コンテンツを表示して、アプリの目的を知ってもらうのもよいでしょう。

ホーム ページ　ホーム ページとは、ユーザーがアプリを起動するたびに表示されるページです。ここに表示されるコンテンツは、目的をはっきりさせ、アプリの特徴がすぐにわかるようにする必要があります。 このページでは、1 番の特徴だけを強調します。アプリの他の部分はユーザーが探し出してくれることを期待しましょう。 ランディング ページでは、ユーザーがアプリの特徴に気付きやすくするのではなく、散漫な印象を与えないようにしてください。

7. デザインのプロトタイプを作成して検証する

設計のやり直しを避けるために、実際の開発があまり進まないうちに、設計内容またはプロトタイプをガイドライン、ユーザーの印象、要件に照らして検証します。それぞれの機能に

ついて、アプリの改善に役立つユーザー エクスペリエンス ガイドラインと、Windows ストアでアプリを販売するために必要なストア要件があります。 Windows アプリ認定キットを使って、Windows ストアの技術要件を満たすかどうかのテストを実行できます。また、Visual Studio のパフォーマンス ツールを使って、すべてのシナリオでユーザーに快適なエクスペリエンスを提供しているかどうかを確認できます。

レスポンシブ デザインとフォームファクター

ストア アプリは、デスクトップ、ノート PC、タブレット、携帯電話で利用できます。アプリは、さまざまなフォーム ファクターをすべて処理できるように設計します。ユーザーはさまざまなデバイスを使い分けながら、入力方式を切り替えたり、画面の向きを変えたり、アプリを切り替えたりします。それに対応して、アプリは移動し、変化し、反応します。

アプリの柔軟なビュー

Windows では、ユーザーに主導権があります。アプリの UI は、ユーザーが使うあらゆるデバイス、入力モード、向き、状況で適切に表示される必要があります。

柔軟なビューを持つアプリを設計すると、アプリの UI は、モニターの向き、モバイル デバイス、入力方法の変更に応じて自動的に再配置します。Windows は、これらの動作を管理します。

横 横向きのビューを設計して、アプリがすべてのフォーム ファクターで適切に表示されるようにします。さまざまなフォーム ファクターが横向きビューにどのように影響するかを考慮します。新しい種類のデバイスでは、画面領域が変更される場合があります。

縦 一部のデバイスは回転することも忘れないでください。縦向きのビューのときのコンテンツのレイアウトを最適化し、できるだけ機能を維持します。横向きから縦向きに切り替わることで、アプリの幅が約 1/3 になります。したがって、縦向きの形式でデザインがどのようになるか考えることが重要です。

複数のアプリを含む柔軟なレイアウト

Windows : サイズ変更できる Windows の柔軟性を取り入れてアプリをデザインしてください。 柔軟なレイアウト ユーザーがアプリのサイズをどのように変更しても適切に表示するために重要なのは、一意の各ビューについてレイアウトを定義し、コンテンツを自動的に配置および配布する柔軟なコントロールを使うことです。

組み込みのグラフィックスのスケーリング

ユーザーが複数のフォーム ファクターでアプリにアクセスできるということは、Windows が動作するすべての画面サイズに合わせて異なる UI を設計する必要があるということなのでしょうか。画面サイズは無数にあります。この疑問に対する答えは、"必要ありません" です。スケーリング機能により、アプリとコンテンツは、7 インチの小さなデバイスでも 30 インチの大きなモニターでも常に適切に表示されます。 必要なのは、柔軟なレイアウトを使い、スケーリング時にアプリのグラフィックスが適切に表示されるようにすることだけです。

詳しくは、「ピクセル密度に合わせたスケーリングのガイドライン」をご覧ください。

入力パターン

タッチ操作に対応したアプリを設計する場合、ペン、マウスとキーボード操作が標準でサポートされます。ユーザーが入力方法を切り替えても、アプリの使いやすさは変わりません。スレートにキーボードを接続しても、問題ありません。アプリは、ユーザーの選択に対して一貫して予想どおりに応答します。

詳しくは、「タッチ操作の設計」と「ユーザー操作への応答」をご覧ください。

デバイス機能

優れたアプリは、実行されているデバイスを最大限に活かします。Windows 8.1 には、こうしたデバイス機能のサポートが組み込まれています。

加速度計やその他のセンサー　最新のデバイスはさまざまなセンサーを備えています。アプリでは、環境光に応じてディスプレイの明るさを調節したり、ユーザーがディスプレイの向きを変えたときに UI を自動的に再配置したり、物理的な動きに応じて処理を行ったりできます。

地理位置情報　標準的な Web データまたは地理位置情報センサーからの地理位置情報を使って、ユーザーの移動や地図上での位置確認、または近くのユーザー、アクティビティ、目的地に関するユーザー通知を支援します。

カメラ　ユーザーは内蔵型または接続型のカメラを使って、チャット、会議、ビデオブログの録画、プロファイルの写真の撮影、周囲の世界の記録など、アプリに応じた作業を行うことができます。

近接通信　ユーザーは、複数のデバイスを物理的に "タップ" することで接続して、複数のユーザーが物理的に近くにいる必要がある状況 (マルチプレイヤー ゲーム) での使い勝手をよくすることができます。

アプリの機能を計画する際は、アプリが実行されるデバイスについて考えます。アプリが適切に動作するためには、特定のデバイス機能が必要か。それとも、その機能がなくても動作できるか。アプリ マニフェストで、アプリがサポートする機能を宣言する必要がありますが、アプリ自体には、オプション機能の代わりを用意することができます。たとえば、旅行用の地図アプリがあり、ユーザーが地図上で移動経路を追跡し、さまざまな場所にタグを付け、コメントを書き込み、ソーシャル メディアに送り、旅の写真やビデオを追加できるとします。地理位置情報は必要な機能ですが、カメラのサポートはオプションにできます。デ

バイスにカメラが付いていない場合、ユーザーは別のデバイスで撮影したビデオや写真をアップロードできます。優れたアプリはすべてのオプションに対応しています。

データのローミング

ユーザーが職場のデスクトップを終了して帰宅した後、自宅のスレートを起動すると、どうなるでしょうか。ファイル、アプリの状態、アプリの設定も自宅に移動します。コンピューターやユーザー セッションが異なっていても、中断したところから作業を始められます。

詳しくは、アプリのデータのローミングに関するページと、「アプリ データの管理」をご覧ください。

Windows Phone のプラットフォームとエコシステム

Windows Phone 向けに記述できるアプリの種類は多岐にわたり、それぞれが特定のユーザーのニーズを満たすことができます。このセクションでは、実現可能ないくつかの種類のアプリを簡単に説明します。

アプリの展望

Windows Phone は、ユーザーの生活を向上する可能性を秘めたソフトウェア プラットフォームです。その能力は、実行できるアプリによって決まります。魅力ある方法で特定のユーザーのニーズを満たすアプリを作ることができるかどうかは、デザイナーと開発者しだいです。

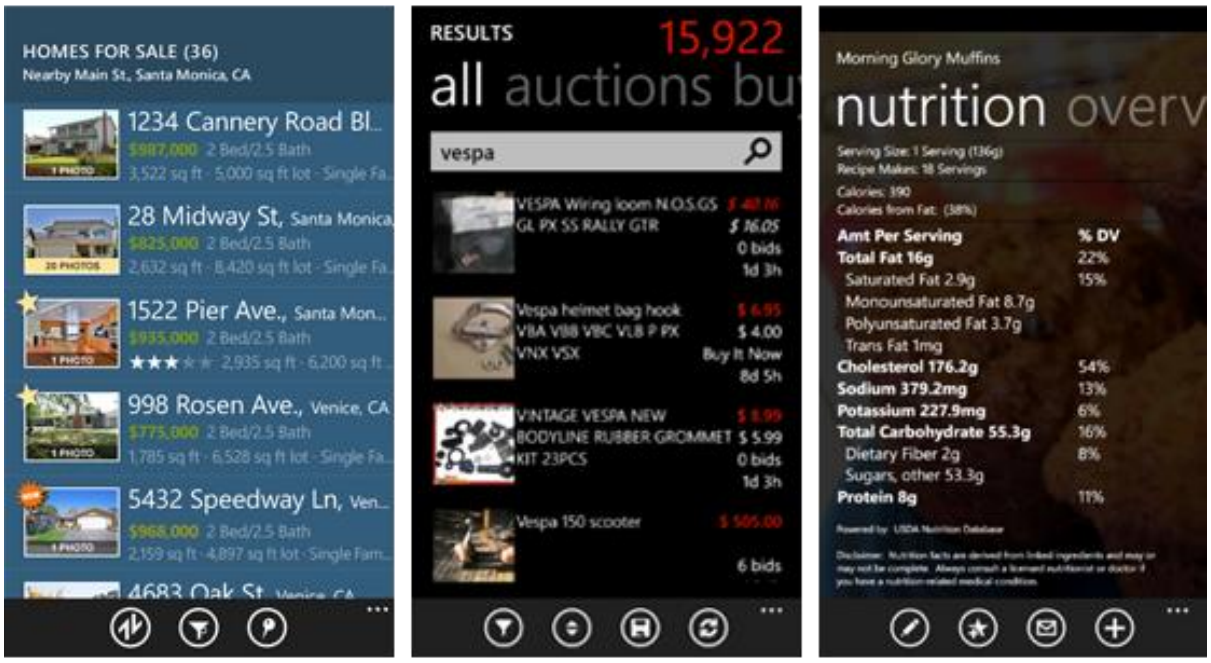
ユーザーは、さまざまな種類の状況とさまざまな理由で Windows Phone を使います。主に友人とつながるために Windows Phone を使うユーザーもいれば、多忙なスケジュールを管理できるように使う人もいます。ゲーム、音楽、ビデオ、読書、ソーシャル ネットワークによって現実逃避を楽しむために Windows Phone を使うユーザーもいるかもしれません。

ストアは、ユーザーがニーズをしっかり満たすアプリを探しに行く場所です。ストアでは、アプリが機能ごとにカテゴリ分けされています。カテゴリによっては、さらにサブカテゴリに分けられています。トップレベルのカテゴリの例としては、ニュースと天気、健康/フィ

ットネス、スポーツ、生産性、ライフスタイル、金融、旅行、ゲーム、エンターテインメント、ソーシャルなど、たくさんあります。ここでは、これらのカテゴリのうちいくつかのカテゴリに含まれるアプリの例について考えます。

ライフ スタイル

ユーザーは、理想的な生活を楽しむことができるように Windows Phone を使います。つまり、ライフスタイルをサポートするアプリを探します。そのような種類のアクティビティをサポートするアプリは、Windows Phone エクスペリエンスの貴重な部分です。

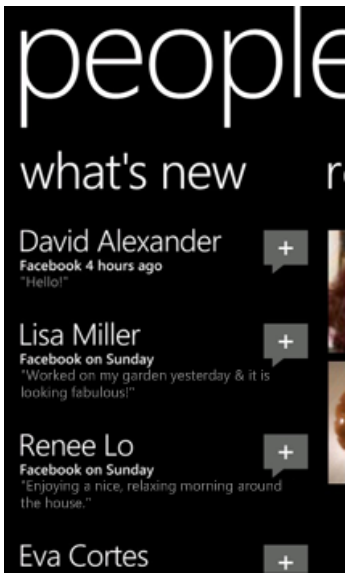


REALTOR.com、eBay、Bigoven のアプリ

これらの種類のアプリは、自分がだれであるか、他の人と何を共有して楽しむことができるかを反映できるため、ユーザーにとって貴重です。上の図に示した 3 つの各アプリは、ユーザーがライフスタイルに合ったものを見つけるのに役立ちます (この場合、パーフェクトな家を見つける、希少な品目を探す、好きな食べ物を料理する)。

ソーシャル

接続されているモバイル デバイスによって、ユーザーに一日中交流できるようになりました。Windows Phone は、人間のこのニーズを満たすのに適したデバイスです。Windows Phone では、ユーザーがテキスト メッセージ、写真、オーディオ、ビデオなどを共有できます。



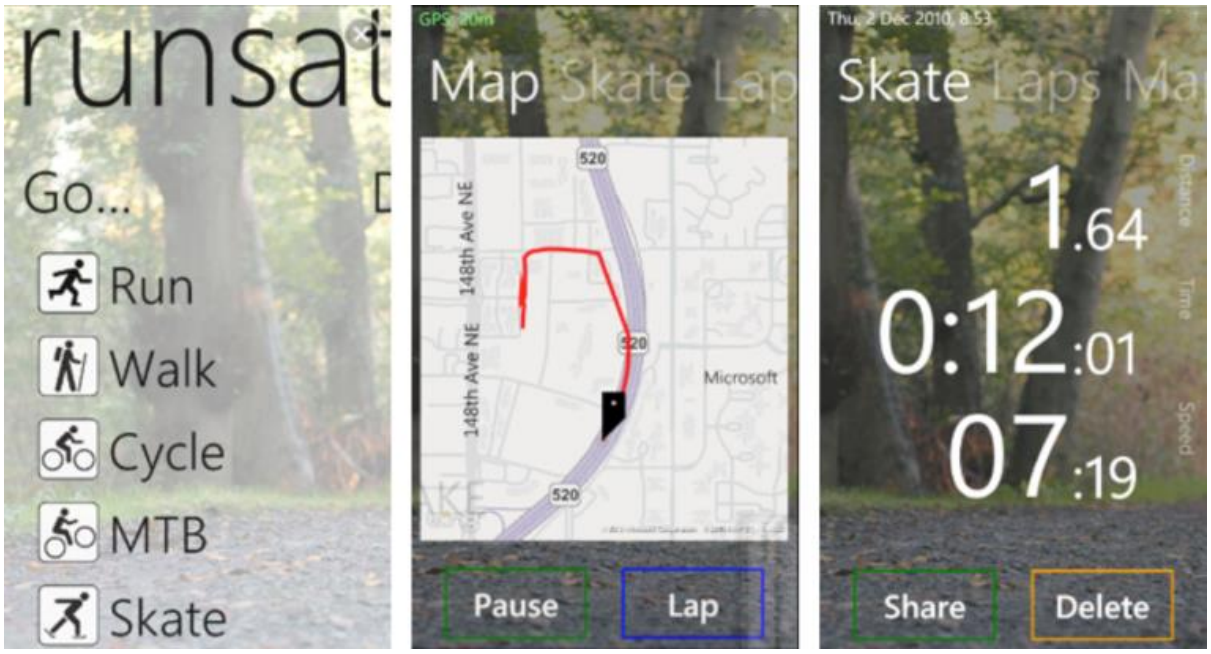
Windows Phone の People ハブ

ソーシャルな交流を可能にするアプリは利用が隔離されていることもあれば、電話の他の機能やアプリと結び付いていることもあります。たとえば、Windows Phone に付属する Pictures ハブは、写真の整理や、ソーシャル ネットワーキング サイトを使って写真のコメントを共有するためにも使われる集約型アプリです。アプリは、写真を変更して写真撮影のソーシャルな側面を強化できるように、写真と連動させることができます。Classic Christmas Cards アプリでは、Pictures ハブから写真を取得し、友人に送信するあいさつ状を作ることができます。

Classic Christmas Cards アプリ

健康 / フィットネス

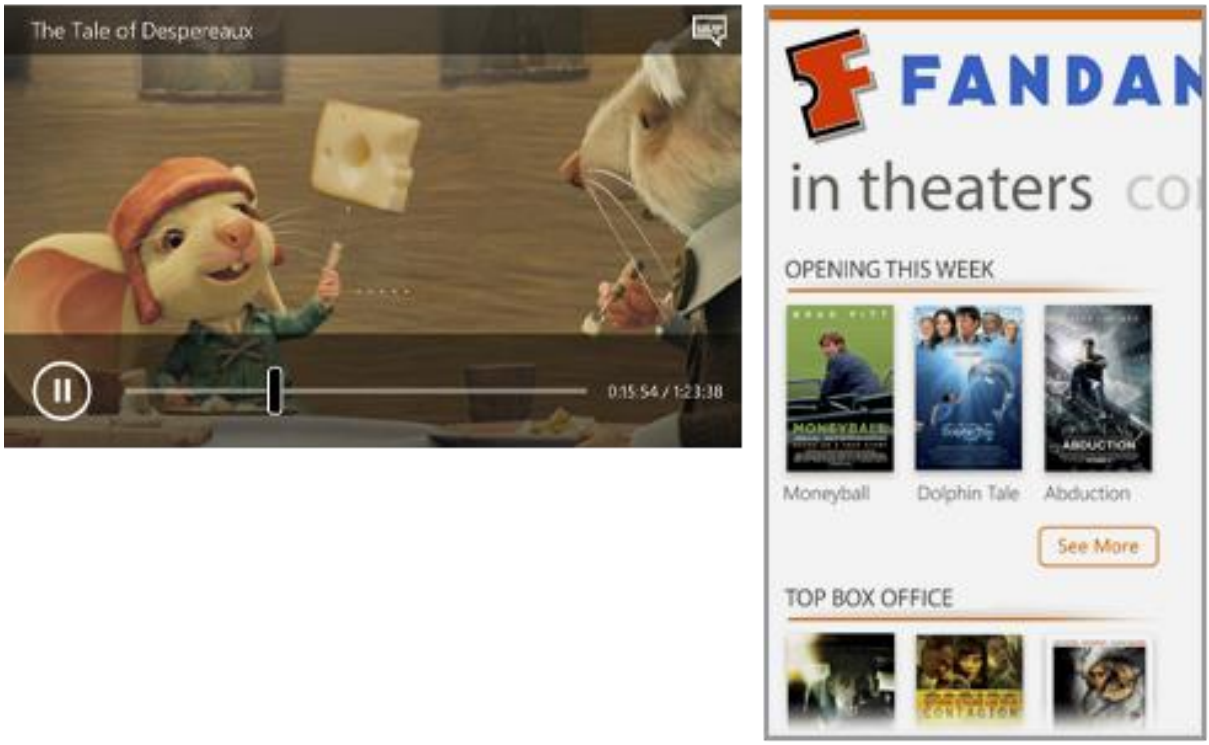
健康/フィットネス アプリでは、Windows Phone のモビリティの側面を利用できます。ユーザーは、電話をジムに持って行き、アプリを使ってトレーニング メニューを行えることをありがたく思います。希望する体のコンディションを入力し、重量挙げや心配療法の指示を得られることを期待します。また、ランナー向けの専門的なアプリでは、ランニング ルートを計画し、走った距離と自己ベスト タイムを記録することができます。



RunSat GPS Sports Tracker アプリ

エンターテイメント

Windows Phone におけるエンターテインメント アプリの可能性には限りがありません。Windows Phone のエンターテインメント性は、洗練されたアプリやシンプルな新しいもの (タッチすると音が鳴る画面上の鈴など) によってもたらされます。Netflix などのアプリは、映画やテレビ番組のストリーミングによってエンターテインメントを提供します。他のアプリは、ユーザーが出席する映画の時間やお笑いショーを見つけることができるようにすることで、実世界のエンターテインメントのニーズに結び付いています。



ビデオをストリーミングしている Netflix と Fandango アプリ

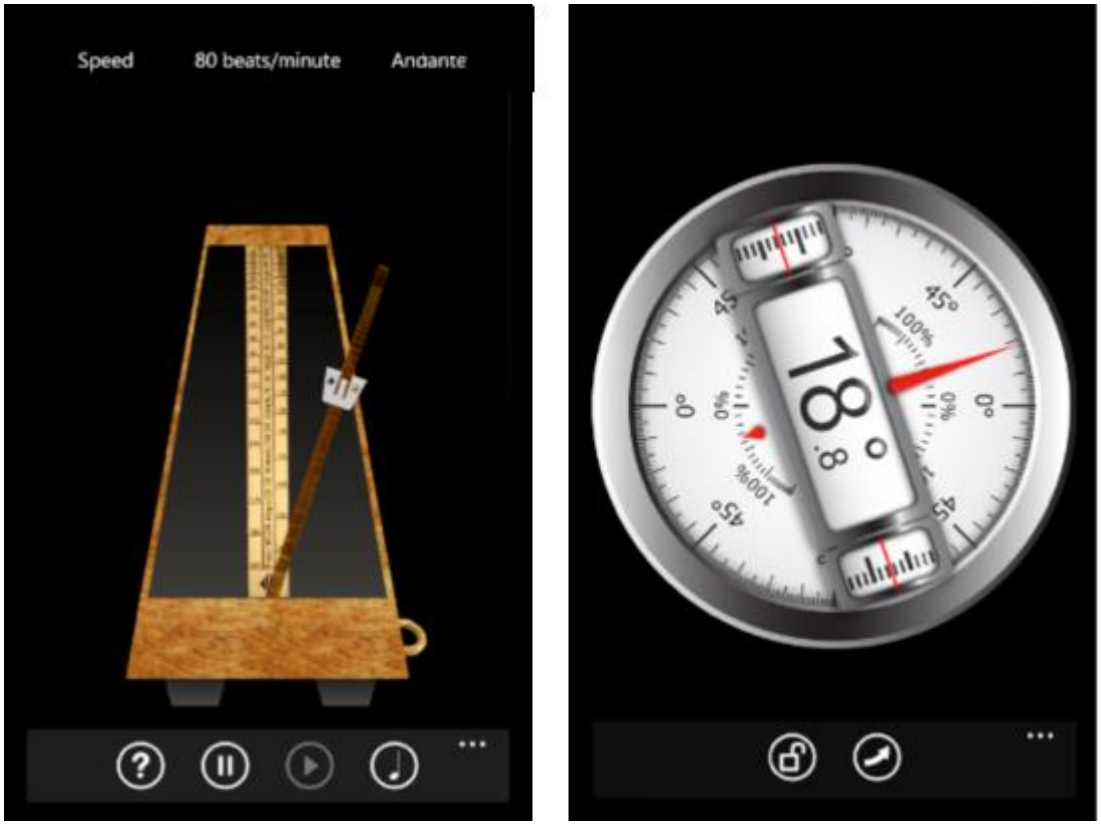
ゲーム

ゲームはいつでもよく売れています。クラシックな Windows のソリティア カード ゲームから、最新の高度な 3D グラフィック アドベンチャーまで、ほぼあらゆる人に向いているゲームがあります。ゲームのカテゴリはかなり大きいため、たくさんのサブカテゴリがあります。Windows Phone は、1 人用ゲームやマルチプレイヤー ゲームをプレイするユーザーにとって適したデバイスです。

ilimilo ゲームと The Harvest ゲーム

ツール

ユーザーは Windows Phones をどこにでも持っていくため、巧みに使えるようにするのはきわめて当然のことです。実世界の装置や道具を模したアプリは、ユーザーの役に立つ可能性があります。シンプルな懐中電灯アプリから定規、メトロノームまで、多くの道具型アプリを利用できます。



Metronome アプリと Clinometer アプリ

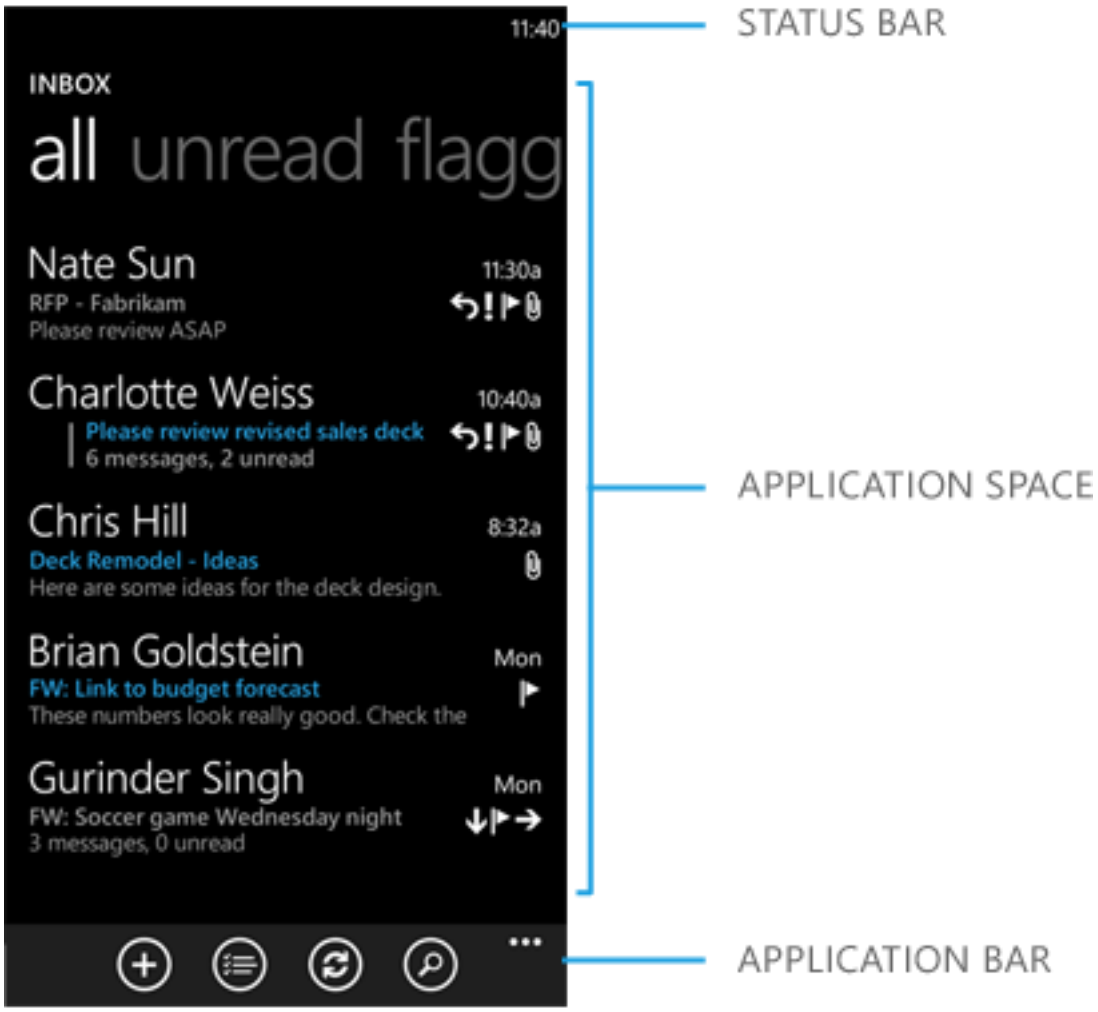
無限の可能性

Windows Phone プラットフォームには、ここで取り上げきれない多くの種類のアプリがあります。ここで説明した事柄は、アプリで実現できる豊かさと信頼性を示しています。Windows Phone 用のアプリの数と種類は時間の経過と共に増加しており、生み出すことができる可能性がすぐに尽きることはありません。アプリは、接続された美しいコンテンツによってモバイル特有の機能をユーザーに提供することを目指す必要があります。

初めての Windows Phone

Windows Phone のユーザー インターフェイス フレームワークは、アプリ ユーザーのための美しくて自然なアプリ エクスペリエンスを作成するために使える一貫したシステム オブジェクト、イベント、操作を提供します。このトピックでは、フレームワークの各部分を紹介し、アプリ ユーザー インターフェイス内での使い方と調整方法についても説明します。

次の図は、あるアプリを実行中の Windows Phone の画面領域を示しています。

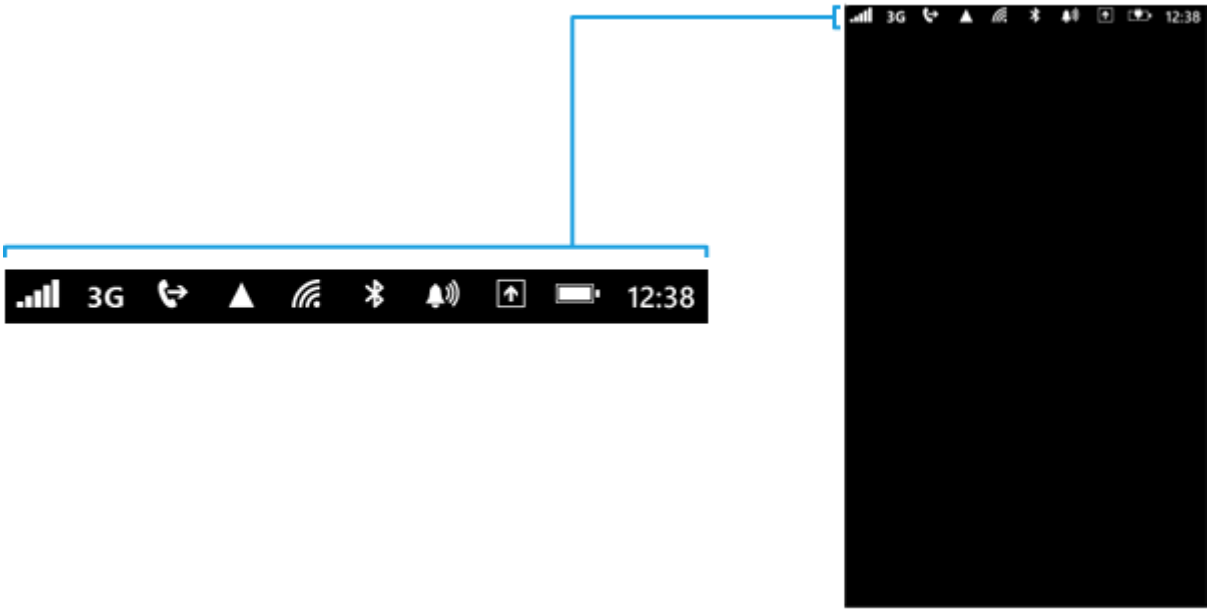


Windows Phone の画面

ステータスバー

ステータス バーは、システム レベルのステータス情報を、アプリのワークスペースの予約された部分にシンプルかつ明確に表示するインジケーター バーです。このバーは自動的に更新され、次の情報を (左から右の順に) 表示することでさまざまな通知を行い、ユーザーがシステム レベルのステータスを常に把握できるようにします。

- シグナルの強さ
- データ接続
- 電話転送
- ローミング
- ワイヤレス ネットワークのシグナルの強さ
- Bluetooth の状態
- 着信音モード
- 入力の状態
- バッテリー電源レベル
- システム クロック



ステータス バー

既定では、システム クロックのみが常に表示されます。ユーザーがステータス バー領域をタップすると、他のすべての関連インジケーターが約 8 秒間表示され、その後画面から消えます。

注 ステータス バーはシステムによって予約されており、変更できません (ただし進行状況インジケーターは更新できます)。非表示にすることはできますが、多くのユーザーはシステム クロックを必須の機能と考えているため、非表示にする際には慎重に検討してください。

ステータス バーは不透明または透明に設定でき、背景色と前景色も変更できます。

アプリ領域

メイン画面領域は、アプリのために予約されています。UI は横モードまたは縦モードで表示できます。ステータス バーが表示されていない場合は、画面全体を使うことができます。

アプリ バー

アプリ バーには、よく使うアプリ タスクを 4 つまでアイコン ボタンとして表示できます。

アプリ バーには、テキスト ヒント付きのアイコン ボタンと、オプションで拡張メニューが表示されます。拡張メニューは、ユーザーが連続するドットのビジュアル インジケーターをタップするか、アプリ バーを上にフリックしたときに表示されます。

アプリ バーは、常に操作ボタン ([戻る]、[スタート]、および [検索]) と同じディスプレイの端に表示され、縦向き、横向きどちらでも画面の幅全体に拡張されます。アイコン ボタンは電話の向きに合わせて回転します。

アプリ バー ボタンは有効または無効な状態で表示できます。たとえば、読み取り専用の項目では [削除] ボタンを無効状態で表示するなどが可能です。

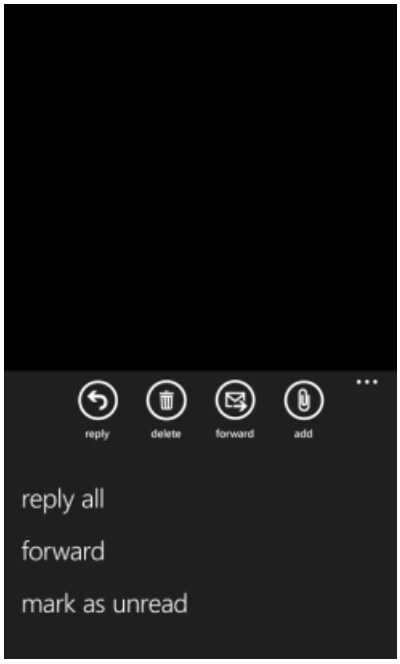
アプリ バーの縦向きでの高さと横向きでの幅は固定で、変更できません。バーは表示または非表示に設定することができます。

アプリ バーは最小化することができ、その場合は高さと幅が変わります。アプリ バーは表示または非表示にできます。透明にしたり、色を変更することもできます。

アプリ バー メニュー

アプリ バー メニューはオプションです。このメニューを使うと、ユーザーがアプリ バーの特定のタスクにアクセスできます。アプリ バー メニューにアクセスするには、アプリ バー内で連続するドットのビジュアル インジケーターをタップするか、アプリ バーを上にフリックします。このビューを閉じるには、メニュー領域外またはドット上をタップするか、[戻る] ボタンを使うか、メニュー項目またはアプリ バー アイコンを選択します。

スクロールせずに済むように、メニュー項目の数は 5 つ以内にしてください。



アプリ バー メニュー

注 表示されるメニュー項目がない場合、アイコンのテキスト ヒントのみが表示されます。ユーザーがアクションを実行するまでアプリ バー メニューは画面上に残ります。

スタート

スタート画面は、ユーザーが電話の電源を入れたときに最初に表示される Windows Phone の画面です。スタート画面には、すばやく起動できるようにユーザーが好みの位置にピン留

めして配置したアプリ タイルが表示されます。電話の [スタート] ハードウェア ボタンを押すと、どのアプリが実行されていても常にこのスタート画面に戻ります。

タイル通知機能により、タイルのグラフィックや、タイルの前面または背面にあるタイトル テキストを更新できます。また、カウンターを使ってユーザーのスタート画面エクスペリエンスをよりパーソナルなものにすることもできます。たとえば、受信したメール メッセージの数、ゲームをプレイする順番が来たこと、現在の天気などを表示できます。

スタート画面は常に縦向きのビューで表示されます。

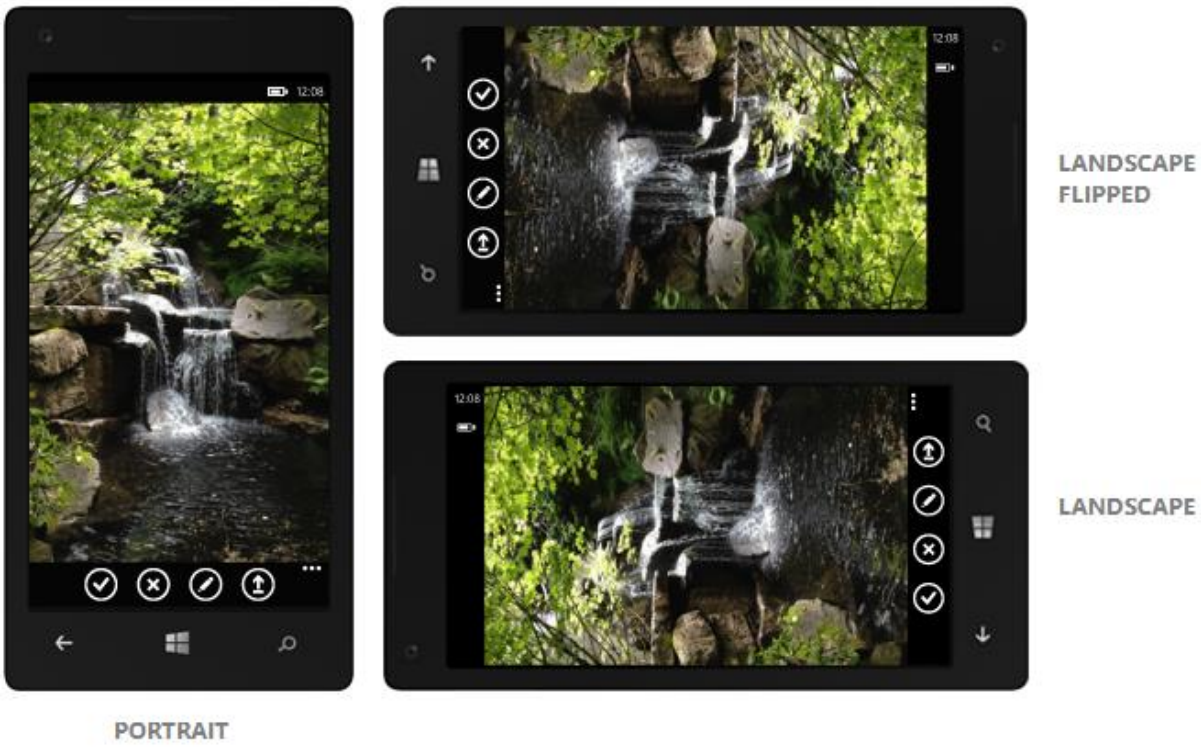
スタート

注 スタート画面は、ユーザーがタイルを配置するために予約された領域です。Windows Phone デバイスには、Microsoft、携帯電話の製造元、携帯電話のサービス プロバイダーによってあらかじめ組み込まれたタイルがあります。アプリでもこの領域にタイルを配置できますが、新しいタイルが配置されるとシステムがスタート画面に移動してユーザーへの通知が行われます。

画面の向き

Windows Phone では、縦、横、反転した縦、反転した横という 4 種類の向きの画面ビューがサポートされています。

縦向きビュー: ページの向きは電話の下部 (反転した縦向きの場合は上部) にある操作ボタン ([戻る]、[スタート]、[検索]) に対して垂直で、ページの縦が横よりも長くなります。縦向きビューはアプリの既定のビューです。



画面の向き

横向きビュー: 横向きビューでは、ステータス バーとアプリ バーが画面の左右に表示されます。横向きの場合はステータス バーが左側に表示され、反転した横向きの場合はステータス バーが右側に表示されます。

どちらの横向きビューでも、ステータス バーのサイズが大きくなります。これは、インジケーターが回転し、最も幅の広いインジケーターが入るようにする必要があるためです。

画面の向きは、次に示すアクションに基づいて変更されます。

最初の画面の向き	回転	変更後の画面の向き
縦向き	60 度左	横向き
縦向き	60 度右	反転した横向き
横向き	60 度右	縦向き
反転した横向き	60 度左	縦向き
横向き、または反転した横向き、テーブル上で水平に配置	30 度上	横向き

向きのプロパティは読み取り専用に設定されているため、プログラムを使って向きを切り替えることはできませんが、固定の向きを設定することはできます。画面の回転が始まると、画面遷移のアニメーション効果が再生されます。

アプリでは静的な向きのビューを定義することも、AutoRotationPreferences プロパティを使って複数の向きをサポートすることもできます。詳しくは、「Windows Phone アプリ内ナビゲーション」をご覧ください。

アプリ内の横向きビューに対応したシステム コンポーネントには、ステータス バー、アプリ バー、アプリ バー メニュー、音量/着信音/倍日レーションの表示、プッシュ通知、ダイアログがあります。

注 画面が縦向きビューのときにユーザーが横向きのハードウェア キーボードを引き出すと、画面の向きがいずれかの横向きビューに変わります。テキスト入力に大きく依存するアプリでは、オプションの横向きハードウェア キーボードを活用するために、横向きと縦向きの両方のモードをサポートする必要があります。

テキスト入力の多い横向きエクスペリエンスは作成しないことをお勧めします。

ハードウェア ボタン

Windows Phone デバイスにはいくつかのハードウェア ボタンがあります。各ボタンには、実行中のアプリを調整したり、実行中のアプリに影響を及ぼす固有の機能があります。正確な位置はハードウェア製造元によって異なります。

1. 電源/スリープ
2. 音量増、音量減
3. カメラ
4. 戻る
5. スタート
6. 検索

[戻る]、[スタート]、[検索] ボタンは、電話の製造元によっては静電式のタッチ ボタンとして実装されていることがあります。

各ボタンが UI に与える影響については、このトピックで後述する各ボタンの説明をご覧ください。

[スタート] ボタン

ユーザーが [スタート] ボタンを押すと、電話のユーザー インターフェイスのスタート画面に移動します。現在実行中のアプリは、イベントを受け取って自動的に一時停止します。

[検索] ボタン

ハードウェアの [検索] ボタンを押すと、Bing 検索エクスペリエンスが開始され、ユーザーはデバイスのどこからでもコンテンツを検索できます。

開発者は [検索] ボタンの動作を変更できません。ただし、アプリで独自の検索ボタンを実装してアプリ内検索を実行したり、Launcher クラスを使って Bing 検索を開始することはできます。

[戻る] ボタン

ハードウェアの [戻る] ボタンは、アプリ内またはアプリ間で前のページ (画面) に戻るために使います。既定ではフレームワークが [戻る] ボタンを処理しますが、アプリでこの動作を上書きできます。また、[戻る] ボタンを使ってメニューやダイアログの終了、前のページへの移動、検索操作の終了、またはアプリの切り替えを行うこともできます。ただし、主な用途は現在のページから前のページに戻ることです。

Windows Phone のページ ナビゲーション モデルについて詳しくは、「Windows Phone のナビゲーション、向き、ジェスチャ」をご覧ください。

重要な注意: アプリが認定に合格して Windows Phone ストアの一覧に登録されるためには、[戻る] ボタンの使用に関連するいくつかの要件を満たしている必要があります。詳しくは、Windows Phone の技術認定要件に関するページをご覧ください。

[戻る] ボタンは、テキスト入力を削除する BackSpace キーとしては使えません。

[音量] ボタン

ハードウェアの [音量] ボタンは、通話の音量 (通話中の場合) や、音楽、ラジオ、ビデオ、アプリ、着信音、システム サウンドなどのデバイス全体の音量 (通話中でない場合) を調整するために使います。

いずれかの [音量] ボタンを押すと、音量コントロールが画面上部のオーバーレイとして表示されます。メディア プレーヤーがアクティブなときは、[前へ] や [次へ] などのオーディオ トランスポート コントロールが [音量] に含まれることもあります。着信音設定のオン/オフを切り替えるコントロールは常に含まれます。このコントロールは、ユーザーが [着信音+サウンド]設定画面で制御できるシステム サウンドの再生に影響します。

電話がロックされていても、メディアの再生中や通話中は [音量] ボタンを使うことができます。

このボタンはシステム全体に作用し、音量設定はアプリにも影響します。このため開発者は、ユーザーが設定した値より音量を上げたり、ミュートを上書きすることはできません。

[音量] ボタンを長押しするとキーを何度も押したのと同じことになり、押したボタンに応じて音量が段階的に大きくなるかまたは小さくなります。

ユーザーが電話を受信したときは、いずれかの [音量] ボタンを押すと着信音が止みます。

バックグラウンド オーディオ エージェントを使うと、どの再生コントロールを有効にするかを変更できます。また、現在再生中のオーディオのタイトルやアーティストを変更することもできます。

開発者は、システムに提供するオーディオ ストリームの音量を、ミュートも含めて制御できます。

[カメラ] ボタン

[カメラ] ボタンは、全押しモードと半押しモードをサポートするデュアル アクション ボタンです。ユーザーが全押しすると、電話でカメラ アプリが起動されます。カメラ アプリの起動後にユーザーが半押しすると、自動フォーカス機能が有効になります。

カメラ アプリ内で [カメラ] ボタンを押すと、カメラ モードのときは写真が撮影され、ビデオ モードのときはビデオ撮影が開始または停止されます。

デバイスがスタンバイ (画面オフ) またはロック状態のときにユーザーが [カメラ] ボタンを 1 秒より長く押し続けると、カメラ アプリが起動されます。

アプリで CameraCaptureUI クラスを使うと、プログラムによってカメラ アプリを起動できます。

未処理のカメラ フィードを使っている場合は、カメラ ボタンを上書きできます。詳しくは、「Windows Phone のハードウェア カメラ シャッター ボタンにアクセスする方法」をご覧ください。

Windows Phone 用の音声

Windows Phone 用の優れたアプリを作成する場合、ユーザーに表示されるコンテンツを考慮するだけでなく、ユーザーへの通知に使う言葉を検討することが重要です。これを音声と呼びます。アプリの音声を改良することで、アプリがユーザーの心に響くようになり、ユーザー エクスペリエンス全体が向上します。

ユーザーに語りかける

パッケージ、製品、Web のいずれであっても、視覚、言葉、音声によって、Windows Phone ブランドのすべての面に反映される信頼性を通じてユーザーとやり取りします。

Windows Phone の音声の原則は次のとおりです。

- 誠実さ
  - 誠意のある直接的な言い方を用います。あいまいさや冗長性を取り除き、わかりやすく要領を得た内容にします。
  - ポジティブな見方を心がけます。問題よりもチャンスやソリューションに重点を置くようにします。
  - 決まり文句や専門用語を避け、自然な句読点と自然な音声を使います。
  - 正確さによって完全性を確保します。ほんの少しの誤りで信頼は失われ、それを取り戻すには何年もかかります。

- 活気
  - 活力と熱意を示します。楽しく魅力的なリズムで表現します。
  - ユーザーにとって便利なアプリになるように、常にユーザーのことを念頭に置くようにします。
  - ユーザーの心を無理につかもうとするのではなく、自分から興味を持ってくれるようにします。
  - 喜びと意義を伝えるように言葉を用います。
- バランス
  - 世界中のユーザーが聞いていることを念頭に、俗語や口語表現の使用には注意します。
  - ユーザーにとってどのようなメリットがあるかを示し、それをプレゼンテーションで強調します。
  - 文脈を意識し、目の前の状況に適した内容にします。
  - 信頼を失う危険があるため、おおげさな約束を避けます。
- サポートになる
  - ユーザーが安心でき、自信が与えられ、成功を期待できるようにします。
  - 解決策やそれを見つける方法を提供するために、勇気を与え、忍耐強く、共感的な内容にします。
  - わかりやすく率直な表現にします。
  - 機能を低下させずにソリューションをシンプルにします。

ファミリールームのテキストは、最初にこの機能を使うときにユーザーを温かく迎えます。

音声の実装

ユーザーを理解し、適切なメッセージを適用し、状況を認識することで、ユーザーに対する最前線に音声を使います。アプリで音声を実装するために役立つヒントを紹介します。

言葉

言葉は、意図やメッセージを伝えたり暗示する働きがあります。単に繰り返すだけではありません。話すだけではなく表示するようにします。アプリのスキャンや読み取りが簡単で、明確な目的と価値を持ち、喜びを与える可能性を示しているなら、それはエクスペリエンス全体に反映されます。

- 「シンプルさ」と「簡単さ」を示すには、シンプルな文章や言い回しを用い、できるだけ短く、日常的な言葉を使います。
- 正確な選び抜かれた言葉は、明瞭さを生み出します。
- 言葉を厳選することで多くを伝えることができます。

技術的な経験が少ないかまったく無いかのようにアプリを眺め、ユーザーのように話すようにしてください。以下に、良い例と悪い例を示します。

- 「提案を見る」ではなく「Windows Phone 8 を入手する」という言い方にします。
- 「無効な ID です」ではなく「customer@microsoft.com のような ID が必要です」という言い方にします。
- 「表示を構成する」ではなく「テーマを選ぶ」という言い方にします。

また、多くのユーザーにとって威圧的に聞こえたり、あいまいな表現になるような、つっけんどんな言い方は避けるようにしてください。例をいくつか紹介します。

- 無効
- エラー
- 構成
- 管理
- デバイス （代わりに「電話」を使います）

句読点

句読点は意味を伝えるのに役立ちますが、言葉の代わりにはなりません。いくつかの重要なポイントを次に示します。

- 感嘆符を使いすぎないようにします。代わりに、より強い言葉を選びます。
- 感嘆符と同様に、疑問符も上手に使います。リンクがユーザーの質問として表示される場合に適しています。

言い回し

ユーザーに語りかけるための適切な音声を見つけるには、口語的な表現 (言葉のあやや、言語特有の言い回し) を使わずに自然に聞こえるようにします。具体的には、次のようにします。

- 短縮形の単語を使うのはおすすめです。形式張った印象がなくなります。
- 短い文章は読みやすいです。
- まずコンテキストが優先されます。ページのデザインは言葉の選択、文章の長さ、句読点などに影響することがあります。

例

悪い例

開こうとしているファイルは、許可された種類のファイルではありません。

良い例

この種類のファイルは電話で開くことはできません。

悪い例

このデモ デバイスの設定を、事前構成済みの値に復元しています。このプロセスは中断できません。

良い例

この電話をデモ用の初期設定に戻しています。この処理にはしばらく時間がかかり、中断することはできません。

悪い例

提供された情報は、アカウントに関連付けられたサービス関連の必須の連絡のため、および興味があると思われる他の連絡のために Microsoft によって使用されます。

良い例

Microsoft は、お客様のアカウントに関するサービス関連の情報、またはご興味があると思われる他の有用な情報をお届けする場合があります。

アプリの設計プロセス（Windows Phone）

このセクションでは、モバイル アプリの設計が Windows Phone の操作性に与える影響について説明します。このセクションで説明する原則の多くははっきりしたものではありませんが、どれも日常的な使用から明らかになったものです。これらの操作性の原則が合わさると、デザイナーが問題を解決し、ユーザーの作業を簡略化し、プラットフォームを活用することができます。

ここでは、コントロールとナビゲーションのプログラムに関する原則や適切な使用については扱いません。

タッチの使用

タッチ操作には、設計に関する新しい考慮事項が導入されました。タッチ デバイスでうまく設計すると、ユーザーはタスクを実行する方法を知ることができ、アプリのブランドが向上します。最良のアプリは、滑らかな感覚でタスクを行うことができるようユーザーを効率的にガイドします。

外見上の設計と対話的操作の設計

アプリを計画するプロセスは、対話的操作の設計と外見上の設計を組み合わせて、反復的な方法で実行されます。対話的操作の設計は、アプリで設計する動作、ジェスチャ、応答の分類を指します。外見上の設計は、色やオリジナルのアートにより、それらの要素を画面上で

生き生きしたものにします。設計のどちらの面も、プログラミングを開始する前に慎重に検討して計画する必要があります。モックアップ、描画、ワイヤーフレームを行って、前もって対話的操作の設計と外見上の設計の両方を計画してから、開発を開始します。プロトタイプについて詳しくは、「最良の Windows Phone アプリを設計する」をご覧ください。

外見上の設計により、ユーザー エクスペリエンスに美しさとブランドを追加できますが、コンテンツから気をそらしたり、ナビゲーションの混乱しないようにしてください。Windows Phone アプリの外見上の設計をうまく行うと、ユーザーがオブジェクトをタッチ、ドラッグ、フリックする場所と方法をさりげなく教えるエクスペリエンスを実現できます。アプリを操作する方法をユーザーが理解できるように、外見上の設計を計画します。

ユーザー対コンピューターのインターフェイスを持つアプリを設計するときのように、ユーザーにタスクをガイドする図形、フォーム、色、コントロールで構成される、魅力的な外見上の設計を生み出すように努めてください。

モバイルについて

経験豊富なアプリ デザイナーであるか、始めたばかりであるかにかかわらず、このガイドから新しい情報を知ることができます。その中には、他のプラットフォームで見られる前提や原則を考え直すことを促すものがあります。高品質な Windows Phone アプリの外見上の設計と対話的操作の設計は、従来のコンピューティングと大きく異なるためです。計画時は、Windows Phone デバイスの操作にできる限り多くの時間を費やしてください。プラットフォームの感触をつかみ、Windows Phone に付属するアプリについて考えて設計プロセスのパダライムとしましょう。

アプリの概念化

このセクションは、アプリの概念化に役立ちます。始める前に、アプリに関するいくつかの一般的な質問を自問し、回答を書き留めます。これにより、重要な情報を後で思い出し、手順を引き返す必要が生じた場合にアイデアの原点までたどることができます。作成を開始する前の設計フェーズでこのセクションを練習として使うと、役に立つ美しいアプリをすばやく作るのに役立ちます。

回答を書き留めたら、アプリの主な画面と機能を簡単に視覚化し、外観を簡単にスケッチします。これは細かい設計を行うフェーズではなく、アプリを概念化する簡単なブレーンストーミング フェーズにすぎないことに注意してください。

計画とスケッチの前に考慮する必要がある質問を次に示します。

- 何を行うアプリにしますか
- アプリの対象ユーザーはだれですか
- どのような立ち位置のアプリにしますか
- アプリはいつどこで使われますか
- どのような種類のコンテンツを表示しますか

何を行うアプリにしますか

この質問には 1 文で答えるようにしてください。どのような機能がアプリの中心となるかを絞り込むのに役立ちます。最初はかなり一般的な観点から考え、まだ具体的な機能には踏み込みませんが、アプリの目的とユーザーが役に立つと感じる理由については明確にしてください。

アプリが実行するタスクと処理をリストしたり、画面に配置するコントロールをリストしたりします。または、アプリを使って何を達成しようとしているかなど、仮定に基づいたユーザーの目標をリストします (たとえば、"e カードを送信する"、"パノラマ写真を撮る" など)。このリストを重要な順に並べ替えます。こうすることで、機能をリストの下方に移動する必要があれば、何を削るべきかがわかります。

アプリの的を特定のタスクやタスク グループに絞り、それらのタスクが明確で価値あるものであれば、ユーザーにとってのアプリの価値はすぐにはっきりします。また、アプリにわかりやすい名前を付けて、ストアでジャンルを選ぶのも容易になります。

いくつかの重要なタスクに集中すると、それらを全面的にサポートすることもできるようになります。アプリの各画面で有効にするオプションや付随機能の簡単なリストを作ります。ユーザーは、アプリ内の必要なものすべてにアクセスできるでしょうか。

ユーザーは、複雑なアプリの内面と外面を学習し続けたいとは思わないかもしれません。タイトル、説明、アプリのタイルからアプリの目的がはっきりわかるようにします。

アプリの対象ユーザーはだれですか

アプリを使うユーザーを想像し、それらのユーザーの好みに合わせて調整することは重要です。あらゆる種類のユーザーの手でたくさんの Windows Phone デバイスが利用されていますが、競争の激しいアプリ市場では、対象ユーザーの範囲が狭いアプリは気付かれない可能性が高い点に注意してください。この段階では、だれがアプリを気に入るかについて先入観を持たないでください。設計プロセスが進むまで特定の対象ユーザーにアプリを絞らないでください。プロトタイプを作り始めたら別の方向からアイデアを取り入れることを選ぶことができます。プロセスの初期に理想的なユーザーを想像して何でも含めようとすると、プロジェクトが複雑になります。

どのような立ち位置のアプリにしますか

ストアにある同様のアプリをいくつか書き留め、自分のアプリと比較対照します。競合するアプリについて読むと、アプローチを変えたくなるかもしれず、まだストアにない新しいアイデアが浮かぶかもしれません。

アプリを掲載するジャンルについてと、他のアプリやサービスを基にして作るかどうかについて考えます。Windows Phone は、Facebook、Google、Outlook、Yahoo! のアカウントに接続して、連絡先、状態の更新、他の情報ストリームを同期しています。外部のソフトウェアやサービスへの接続は、Windows Phone アプリをプライベートなものにすることに貢献します。

適切であれば、サード パーティのサービス (特にソーシャル ネットワーキングやエンタープライズ ソフトウェア サービス) を使って、ユーザーが電話から Web に友人やコンテンツを広範囲に出し入れできるようにすることを検討してください。これにより、電話で作ったり使ったりした作品とメディアを、友人や同僚のネットワークと共有できるようになります。

アプリはいつどこで使われますか

たとえば、アプリが "ジム" や "車" で使われるとします。これらのシナリオによくある気をそらすものについて評価し、アプリでユーザーによる単純な間違いを防ぐ方法を書き留めます。最高品質の Windows Phone アプリでは、ユーザーの間違いを予想してそれを補う設計が使われています。ボタンを大きくしてタップしやすくしたり、片手で使いやすいようにコントロールをユーザーの親指の下に配置するなど、シンプルな方法のこともあります。

どのような種類のコンテンツを表示しますか

コンテンツはアプリの中心と言えるものであるため、アプリで強調されるコンテンツと、モバイル デバイスでどのように表示されることをユーザーが望んでいるかを検討してください。たとえば、天気予報アプリでは、地域の現在の気象条件を示す画面上のアニメーションを表示することができます。

興味を引く動的なコンテンツがあると、ユーザーはアプリを何度も使いたくなるかもしれません。画面にコントロールを多く配置しすぎて気をそらすことがないようにしてください。ユーザーがアプリを初めて試す場合、好奇心からほとんどのコントロールを操作したい衝動に駆られるものです。各コントロールの機能と効果をすぐに知ることができるように、画面条には少数の十分なコントロールを配置してください。

可能であれば必ず、コンテンツを表すコントロールではなくコンテンツ自体をユーザーが操作できるようにするという設計のアドバイスに従います。コンテンツを処理するコントロールを配置するのではなく、ジェスチャに反応するコンテンツ自体を配置します。たとえば、音楽アルバムのイメージがある場合、再生するための別個のコントロール ボタンは必要ありません。アルバムのイメージをタップすれば再生されるようにする必要があります。スペ

ースを無駄にするコントロールがなくても、どのようにイメージをつかんで拡大できるかを考えます。

決して使うことがない広範な機能がアプリにあるという印象をユーザーに与えないでください。他のアプリのナビゲーション、操作、コントロールについて調べます。その設計から学ぶことができる教訓はありますか。目的に合わせてどのように改善したり、修正したりすることができるでしょうか。場合によっては、余分な機能に時間と労力を費やす前に、アプリの簡易バージョンを配信して市場に出たらどうなるかを見ることは戦略上有利です。

アプリはどのようにハードウェアを使うことができますか

Windows Phone デバイスの機能に精通し、ブラウザーに表示される Web アプリ以上の価値をアプリが加える必要があることを念頭に置いてください。ユーザーは、アプリが電話の加速度計、カメラ、GPS、光センサー、マイク、スピーカーを賢く適切に使うことを評価し、それを期待することもあります。

4 つのタッチポイント

ユーザーに通知を行って注意を引き付ける、次の 4 つの重要な表示チャンスについて考えます。

- ホーム画面
- 主要なコントロール
- 美しいビット
- スタート ページのタイル

ホーム画面

ユーザーがいつ初めてアプリを起動したいと思うかについて考えてみましょう。前の手順でリストしたユーザーの目標とタスクを見直します。アプリを開いたときに新しい情報が表示されるでしょうか。

主要なコントロール

アプリの最も重要な画面に表示するコントロールの一部をスケッチします。どのような種類のタスクをグラフィックで表示しますか。何を活字で表示しますか。タップが頻繁に必要なアプリの場合、想像しているコントロールの位置が手で操作しやすいかどうかを考慮してください。

美しいビット

ユーザーは、最新のモバイル アプリが視覚的に美しく、よく考えられたレイアウトであることを期待しています。アプリは、意図的なオープン スペース、活字、アニメーションを見た目を良くすることができます。または、ミニマリズムな設計にし、コンテンツを中心に持ってくることもできます。何を決定したかにかかわらず、ユーザーの注意を引くようにしてください。

スタート ページのタイル

ユーザーがアプリをスタート ページにピン留めした場合、ライブ タイルを使うチャンスがあります。ライブ タイルは、自動更新され、アニメーションを使って新しい情報が表示されます。電話の画面に小さいビルボードとして表示されるアイコンについて考えてください。何をユーザーに伝えるのでしょうか。どのようにアイコンと色を使って、アプリの目的を簡単に伝えることができるでしょうか。

見栄えのするタイルの作成について詳しくは、「タイルとバッジのガイドライン」をご覧ください。

簡略化と設計

概念を簡略化することは正式なプロセスではなく、"簡略" という用語は定義が困難です。しかし、アプリをカプセル化するアイデアをいくつか書き出したら、複雑さを軽減するためにそれらのアイデアを見直してください (プログラミングを簡単にしたり、アプリを使いやすくしたりするため)。時間を取ってすべてのアイデアを見直します。効率は向上しますか。削除できる機能や他の機能と結合できる機能はありますか。

ここでは、初期設計を考え出しました。アイデアをテスト可能な実際のプロトタイプに変える方法として、次のトピック「最良の Windows Phone アプリを設計する」をご覧ください。

最適な Windows Phone ストア アプリの設計

このでは、 Windows Phone プラットフォームに統合された外観を持つアプリの設計について説明します。このトピックおよびその他の設計プロセスのトピックで説明する設計の原則と戦略は、拡大するストアの中でアプリを際立たせるための鍵になります。また、設計フェーズ中にこうした原則と戦略に従うことで、反復作業時に見つかる操作性の問題の解決が容易になります。高品質の設計では、優れたカスタマー エクスペリエンスと優れたレビューに対応します。これは、競合アプリに対してアプリを差別化するうえで重要です。

ブレーンストーミングと計画については、「Windows Phone 用のアプリの概念化」を参照してください。

設計プロトタイプの主要な側面

プロトタイプの設計では、それまで行ったすべてのブレーンストーミングをまとめます。このセクションでは、アプリの外観のモックアップをスケッチし、設計の重要な側面をテストするために使う方法について説明します。

注 プロトタイプを作成することは、品質向上だけでなく、作業時間とフラストレーションの軽減につながります。このセクションのアドバイスに従うことで、より少ない時間でより優れたアプリ設計を作成するために役立ちます。

Blend for Visual Studio は、現実的で迅速なプロトタイプの作成に役立ちます。Blend により、モックアップをスケッチし、対話的なプロトタイプを作成できます。対話的なプロトタイプは、ユーザーに理解しやすいエクスペリエンスにするためにカスタム UI、アニメーション、切り替えを必要とするアプリ領域で役立ちます。この対話式操作は、多くの場合にアプリの成功にとって不可欠であり、ユーザーが "直感的" な操作性テストを行うことを可能にします。また Blend では、プロトタイプから直接、実際に動作する設計を作成できます。Blend については、Microsoft Expression のホーム ページをご覧ください。

Microsoft Visual Studio 2012 で利用できる PowerPoint Storyboarding は、スケッチのために使うことができるもう 1 つのツールです。このツールにより、事前定義されたストーリーボードの図形のコレクションからストーリーボードを構築し、既存のユーザー インターフェイスをキャプチャできます。詳しくは、PowerPoint を使ったユーザー ストーリーまたは要件のストーリーボードに関するページをご覧ください。

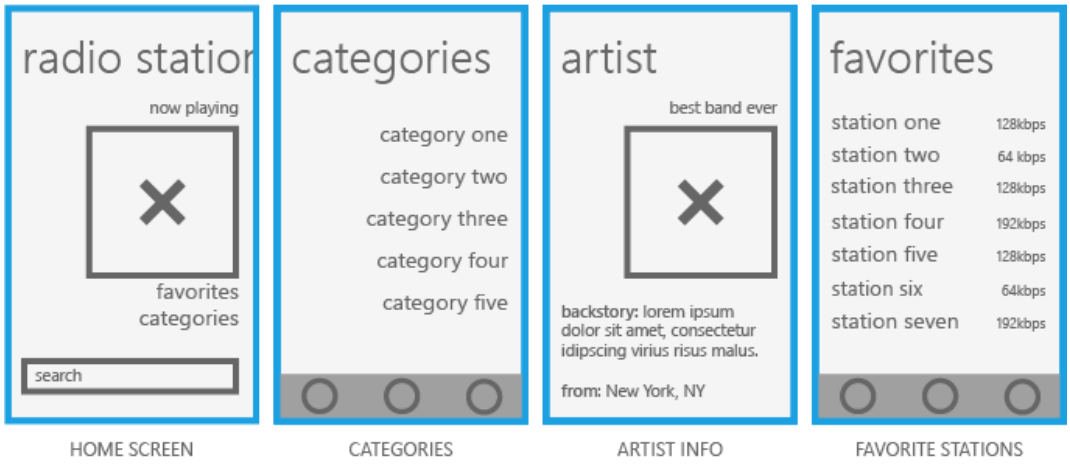
概念のスケッチ

このセクションでは、アプリのモックアップを作成する方法について説明します。

設計プロトタイプ作成の第 1 段階では、まず鉛筆と紙から初め、すぐに PC でのスケッチに移行します。モバイル デバイス用のアプリのスケッチは、デスクトップ ソフトウェアの場合とは異なります。ここでは、こうしたアプリのナビゲーション階層は浅く、スコープは狭いために、ストーリーボードはそれほど役立ちません。モバイル UI の課題は、直感的かつ動的な方法で操作とデータを提示することです。

アプリにエキスパート タスクが含まれる場合、可能な限り少ない (かつ簡単な) 手順になるように、タスク個別のストーリーボードを作成します。次に、汎用ストリーミング ラジオ アプリの一連のプロトタイプ スケッチを示します。

1. ユーザーが最もよく表示および使用する静的な画面をスケッチします。たとえば次の図は、下のストリーミング ラジオ アプリの予備的な描画を示しています。ホーム画面、カテゴリ、アーティスト情報ビュー、お気に入りのラジオ局のリストのモックアップがあります。



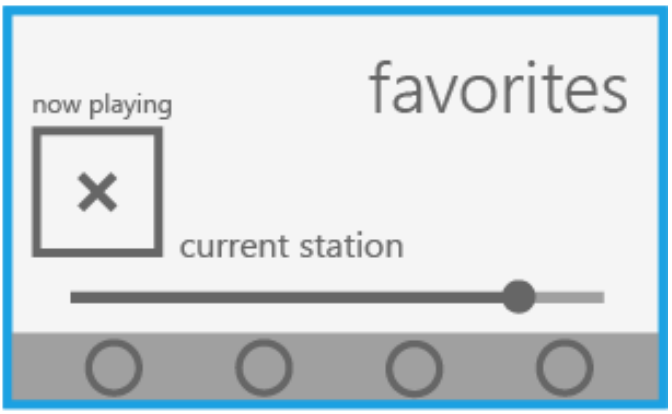
2. アプリがユーザーに特定の一時的状態を示す場合は、その様子をスケッチします。この例では、音楽プレーヤー アプリを構築している場合に、設定とスプラッシュ スクリーンのビューを描画します。

LOADING/SPLASH SCREEN

いくつかの状態

3. カスタマイズする可能性があるすべての重要なコントロールをスケッチします。画面の分類を簡単にするために、どのようにカスタム コントロールを使用できるかを考えます。このモックアップでは、再生のための詳細なコントロールと、コントロールを含む特別な横向きのビューを示します。

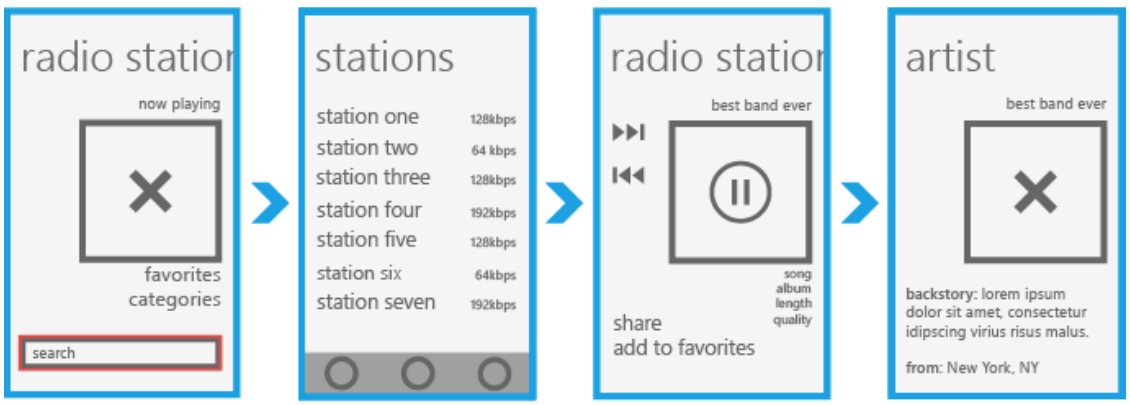
NOW PLAYING SCREEN



FAVORITE STATION LANSCAPE CONTROLS

重要なコントロール

4. ここで、タスクを実行する一連の手順として、いくつかの画面をまとめることができます。重要なタスクまたは操作を完了するためにユーザーが必要とする手順を計画します。タスクの各手順に関連する静的な画面を順に描画します。次の図は、アーティスト情報を表示するまでの検索操作のフローを初めから終わりまでマップするモックアップを示しています。

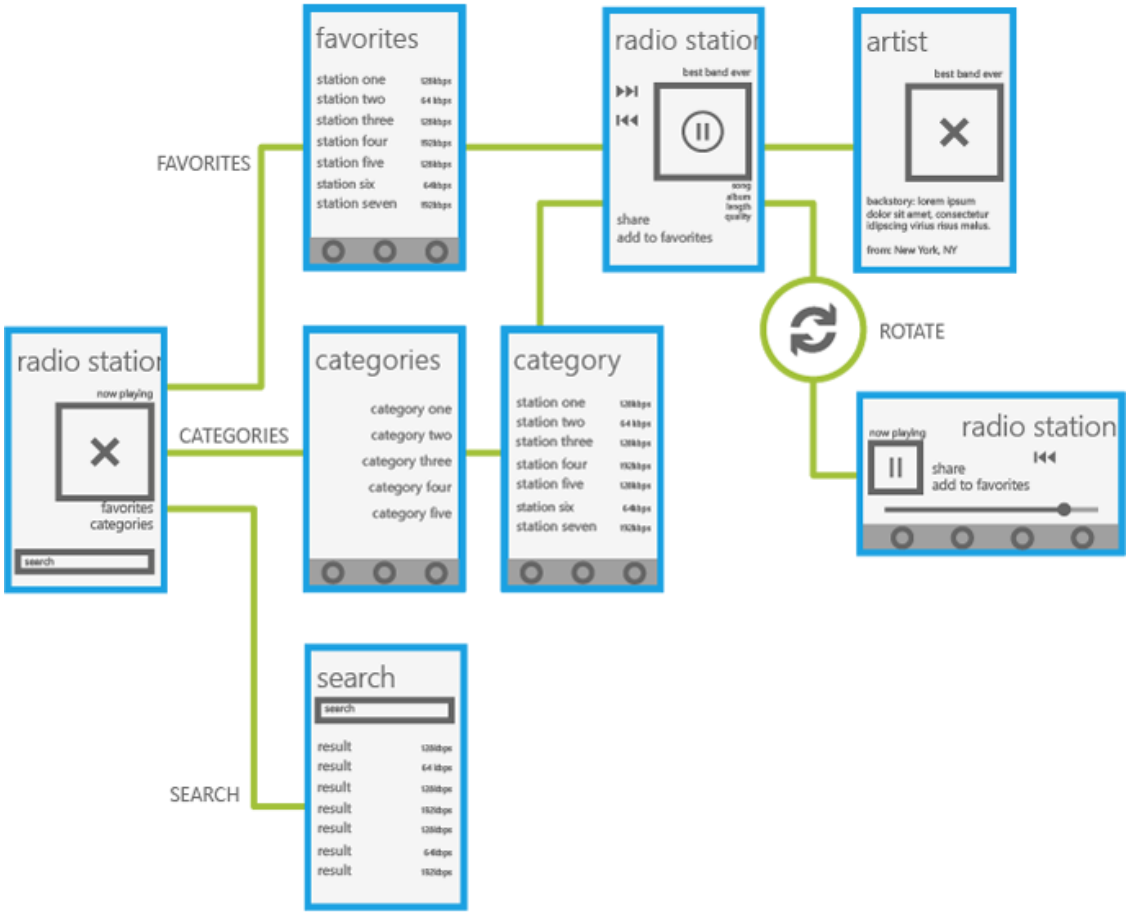


HOME SCREEN　SEARCH RESULTS　NOW PLAYING SCREEN　ARTIST INFO

一般的な検索タスク

この手順を実行するには、まずユーザーがアプリにより実行するすべてのタスク (音楽ジャンルの検索、お気に入りリストへの新しいラジオ局の追加など) を挙げる必要があります。次に、識別された各タスクのマップを描画します。これらのマップでは、指定のタスクを実行するための画面のパスを示します。

5. アプリの全部分の静的な画面間の完全な関係からスケッチを試みます。ユーザーがタスクを進行している感覚が得られる方法で、関連する画面の外観と機能を作るようにします。次の図に、簡単なストリーミング ラジオ アプリの完全な情報アーキテクチャ マップを示します。



マスター マップ

マスター マップはタスク マップとは異なります。静的なマスター マップでは、特定のタスクを実行するフローは示さず、各画面および画面間の階層関係のみを示します。前の例では、ホーム画面から、お気に入り画面、カテゴリ画面、検索結果画面につながることがわかります。

数枚の図面をピン留めした後、Blend for Visual Studio で新しいプロジェクトを開くことによりプロトタイプを作り始めます。

注 Blend によるプロトタイプの作成時には、付随する XAML マークアップが内部で自動的に記述されます。これによって、反復処理でレイアウトとスタイルを作成できます。こうしたアーティファクトは、おそらく忠実度が低いものですが、運用アプリと同じ書式であり、したがって互換性を持ちます。

プロトタイプまたはモックアップの重要な領域の識別

モバイル アプリでは、特定のフォーカス領域を選択することにより、迅速にプロトタイプを作ることができます。一般に、アプリのすべての側面について詳細なレベルでプロトタイプを作る必要はありません。最も重要な側面、または最も困難な側面を選択し、そこにプロトタイプのフォーカスを合わせます。設計者は通常、新しいテクノロジ機能、または開発チームの通常の専門知識から外れるすべての側面について、優先順位を付ける必要があります。

大半のアプリについてこれは、ユーザーに価値を提供するモックアップ タスクまたは操作を意味します。アプリに対して構想した目的を記述することを考慮します。発生することが想定される出来事を手順をおってプロトタイプを作成し、カスタム コントロールによって手順を減らすことができるかどうか、またはそれ以外の方法で迅速化できるかどうかに着目します。

一般的に、Web または別のプラットフォームのアプリを Windows Phone に移行する場合、アプリ画面のコントロールとグラフィックスを減らし、すっきりと整理することが必要になります。また、合理的なある分類に属する複数の画面にタスクを分散し、Web では大きなグループ内にコントロールをまとめることが必要になる可能性があります。アプリのグラフィックス、コンテンツ、写真を拡大する必要があるかどうかを確認します。ネイティブ コントロールと、可能であればいくつかの WebView コントロールによってプロトタイプを作ります。

実際的な反復回数を前もって決定しておきます。

プロトタイプの詳細レベルの決定

プロトタイプを迅速に作成し、構築に移行するために、反復処理で最優先とするアプリの要素を決定します。ここでは、アプリの次の部分のプロトタイプをどの程度詳細に作るかを決定することも含まれます。

- 表示の詳細レベル

  このアプリの関係者にとってアートと外観が最優先である場合は、特徴、機能、画面、状態を示す一連の詳細な図面を作ります。またカスタム コントロールとその状態をモックアップする必要があります。

  注 Blend for Visual Studio でカスタム コントロールまたはユーザー コントロールをモックアップすることで、運用コントロールの構築に役立つ XAML マークアップが得られます。

  表示詳細モックアップには、以下を含める必要があります。

  1. スタイルとテーマ
  2. スプラッシュ スクリーン
  3. アプリのタイル

- 機能の詳細レベル

  機能または特徴がアプリの中心的な役割である場合、対話的および半機能的なプロトタイプを構築した方がよいこともあります。静的な画面または静的な画面グループから始め、その画面間の操作を 1 つずつ構築します。機能の効率を高め、コントロールを減らし、必要な入力を軽減する方法を考慮します。

- コンテンツの詳細レベル

  アプリの関係者がコンテンツの提示に主に関心を示している場合は、UI の構築時に可能な限り最も正確なプレースホルダーを作成する方法を決定することを検討した方がよいケースもあります。コンテンツがテキストである場合は、文字体裁に細心の注意を払って設計をモックアップし、コンテンツの配置を表現するためにブロック、その他のアブストラクションではなく充てんテキストを使います。

  アプリでビジュアル メディアを提示する場合、元のプロトタイプ プロセスに紙と鉛筆の作業を追加した方がよいこともあります。再生中に表示されるようにアプリのコンテンツの詳細なスケッチを描画し、横および縦の両方向の図面を含めます。向きが変更されたときには、コントロールのカスタマイズまたはレイアウトの変更を忘れずに考慮します。

アプリのローカライズを計画している場合は、ここで各言語に必要な十分な余地を残し、グローバリゼーションの問題に対応する準備をします。ローカライズとグローバリゼーションの問題を理解するためのその他の参考情報も用意されています。

- ブランドの詳細レベル

  商業的な目的でアプリを構築しており、ブランドが管理されている場合は、詳細まで細心の注意を払った包括的でブランド化された視覚的なプロトタイプを作ります。これによって、企業の標準の色、レイアウト、およびロゴから最終製品が逸脱していないことを確認できます。

一貫性を保ったカスタマイズ

カスタム視覚要素によって、アプリの品質、独創性、ブランドを印象付けることができます。アプリのビジュアル デザインのカスタマイズ量を事前に決定し、それにしたがって要素を計画します。カスタム アートを導入し、アプリ設計プロセスで後ほどブランド化することは難しい場合があります。

カスタマイズする必要がある要素には、テーマ、テンプレート、コントロール、スタイルがあります。

Windows Phone で視覚要素をカスタマイズする場合は常に、次のガイドラインに従います。

- リアリズムよりも実際的な操作を優先します。

  一般的に言えば、アプリのカスタム コントロールでは、現実を模倣することは避ける必要があります。たとえばラジオ アプリでは、ラジオ局の選択を制御するためにダイヤル、ノブ、一連のボタンを使うことは避けます。代わりに、周波数を調整するスライダー コントロール、ラジオのオンとオフを切り替える再生/一時停止ボタン、事前に設定されたラジオ局を保存するお気に入りボタンを使います。

FM ラジオ アプリ

実用性のために、アプリが全画面表示モードでコンテンツまたはゲームプレイを表示する場合に、一部またはすべてのコントロール、アプリ バー、ステータス バーのフェード アウトや削除行うことも有効です。

- リアリズムよりも実際的な操作を優先します。

  カスタム コントロールを含むアプリの一貫性と信頼性を確保するために、記号、図形、単語、色が一貫した意味を持つ基の設計スタイルをアプリに含めるようにします。

  相互に影響する相関カスタム コントロールのグループを示すことを計画している場合は、意図した方法であらゆるコントロールの組み合わせが合理的に動作することを確認します。

  可能である場合は、すべての考えられる操作が合理的であることを確認するために、カスタム コントロール間の操作を描画します。ユーザーに混乱を招くシナリオが強制される組み合わせでは、代替の設計を使います。

操作性のテスト

これで、プロトタイプがさらに詳しく定義されたので、設計を中断し、Windows Phone 設計のパラメーターが範囲外でないことを確認します。次の質問の回答を得ることにより、デバイスでプラットフォームと OS を最大限に活用しているかどうかを確認します。

プロトタイプに関する確認の質問

- アプリ設計で、Windows Phone ハードウェアの採用方法は明確ですか。
- アプリのタスクでは、操作は快適ですか。
- アプリであまりにも多くの情報または機能を提示していませんか。情報が少なすぎることはありませんか。
- アプリで、ユーザーにとって価値があると考えられるタスクまたは操作を行うことができますか。
- 別のモバイル プラットフォームからの移行ですか。Windows Phone ユーザーは、より少ないタップ、明瞭なビュー、大きな文字体裁、コントラストと色分けを期待します。
- 両軸 (X 軸と Y 軸) および両方向 (縦向きと横向き) のスクロールを使っていますか。アプリの目的によっては、ユーザーはその両方を期待します。
- 指先で簡単にコントロールを操作できますか。コントロール テキストは明瞭ですか?
- UI で、タッチと進行状況のフィードバックを提供していますか。
- ハードウェアの戻るボタンの使用を考慮しましたか。
- 埋め込み Web コンテンツ (および WebView コントロール) の使用は控え目ですか。
- アプリで、全画面表示のコンテンツ再生中にステータス バー、コントロール、アプリバーをフェード アウトしていますか。
- Pivot コントロールおよび Hub コントロールを効果的に正しく使用していますか。

カスタム UI 要素のテスト

カスタム UI 要素をモックアップしたら、自身でテストするようにします。Blend for Visual Studio で作業している場合は、(Behaviors の機能を使って) ページまたはコントロールの動作とその視覚効果をモックアップできます。

ユーザーとの協議

プロトタイプを作り始めたらすぐに、アプリの対象ユーザーから成る信頼できる小グループを見つけます。満足できたらすぐに、信頼できるユーザーにプロトタイプを提示し始めます。想定されている操作を説明しないでください。その代わりに、ただ待機し、ユーザーがアプリの動作と操作方法をどの程度簡単に理解できるかを確認します。

注 一般的に、ワイヤーフレームの図面ではなく、以前のユーザー テストの視覚的詳細を含むモックアップを使います。ワイヤーフレームの図面では、ユーザー エクスペリエンスが正確にシミュレーションされません。

ユーザーへの早期の伝達

ビジネス向けアプリを設計している場合は、ターゲット ユーザーに加えて、管理者と非専門家のニーズと優先順位をテストするようにします。たとえばプロジェクト管理ソフトウェアは、プロジェクトで作業するユーザーに加えて、プロジェクトの進行状況を確認するためにアプリを使うその上司にとっても有用です。

注 ますます多くの機能を求めるユーザー要求の犠牲にはならないでください。一部のユーザー フィードバックが重大な UI の問題を警告することもありますが、一般に大半のユーザー フィードバックは総体的に扱う必要があります。アプリのフォーカスを維持し、最小限に抑えます。ユーザーには、UI や操作について個々に示唆を提供するよりも、アプリの目的を明確に伝える方が良い場合が多いことに注意してください。

最終モックアップの作成

設計プロセスのこの時点でアプリは、以前に説明した Windows Phone の操作性のガイドラインを既に満たしています。最終モックアップを評価する場合は、次の手順を実行します。

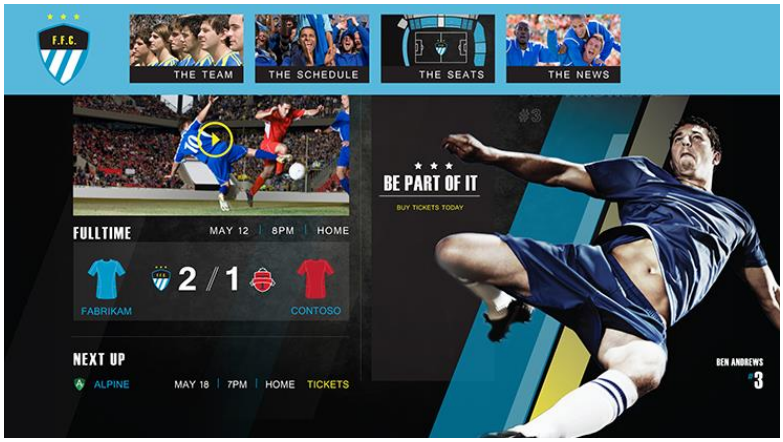
1. プロトタイプがアイデア収集時に計画したニーズと標準を満たすかどうかを判断します。
2. 次に、モックアップが最終的なアプリ レビュー プロセスによって必要とされる詳細レベルを満たしているかどうかを判断します。「設計のプロトタイプの主要な側面」を確認した後、プロトタイプで必要とされると判断した詳細レベルを思い出してください。

3. 最終モックアップには、次の適切なレベルの詳細を含める必要があります。◦視覚要素ー文字体裁とコンテンツが明確、明瞭、簡潔に示されていますか。アプリに視覚的な魅力がありますか。
   - 機能要素ー目的と操作の両面で直感的なタスクですかアプリの処理、およびアプリの操作方法は明確ですか。
   - コントロール要素ーすべてのカスタム コントロールの機能が一貫性のある内部設計言語に従っていますか。容易なタッチ操作を可能にするサイズ設定と配置ですか。
   - ブランド要素ー色とロゴが正確に複製されていますか。すべてのアートが著作権の条項に準拠していますか。
4. プロトタイプの作成中に明確にした操作が維持されていることを確認します。最も一般的または重要なタスクおよび操作の外観とフローが意図するものであることを確認します。
5. 設計の最終処理を行い、運用を開始したら、提出のための設計要件を満たすアプリの準備を始めるために、「Windows Phone 用の初めてのアプリを作成する方法」をご覧ください。

独創的なビジュアル

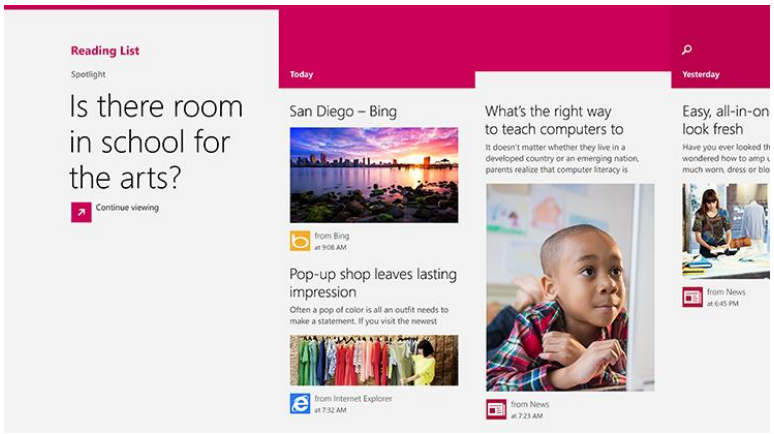
Windows ストアアプリは、ユーザーに使われることを意識してデザインしなければなりません。美しく独創的にデザインするために、さまざまな要素を検討することが必要です。

ブランド



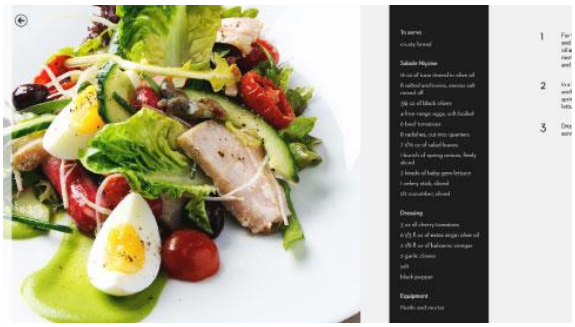
Windows は、ブランド価値を高めるフレームワークを提供しています。フレームワークでブランドを表現する方法を確認してください。

レイアウトとコンポジション



Windows はいろいろな方法でグリッドを並べ替え、多様な固定レイアウトを作り出します。

アニメーション

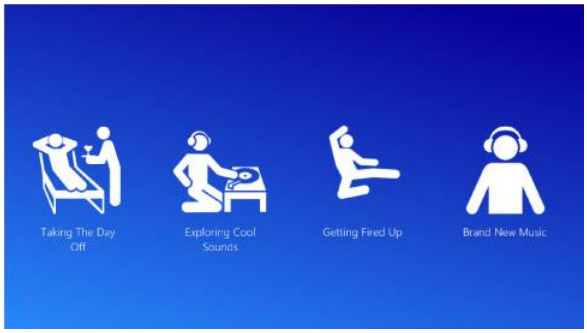


アニメーションにより、アプリに快適さや楽しみが付加されます。アプリに最適なアニメーション化の方法を見つけてください。

書体

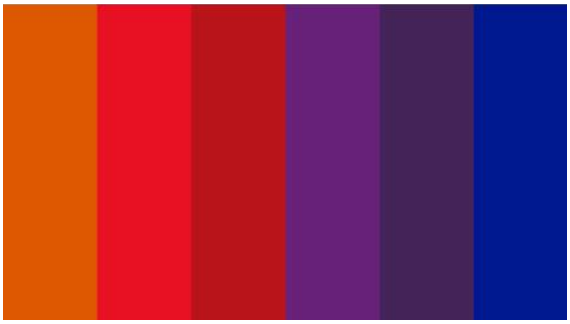
書体によりブランド価値を高めます。階層と組織を書体で示すことができます。美しい書体の可能性をご確認ください。

アイコン



アイコンは、その簡略さとスタイルにより、瞬時に意味を示します。使いやすさのために、標準または独自のアイコンを使います。

色



色を使って、ブランド、個性、目的を表現します。アプリの色でストーリーを印象づけます。

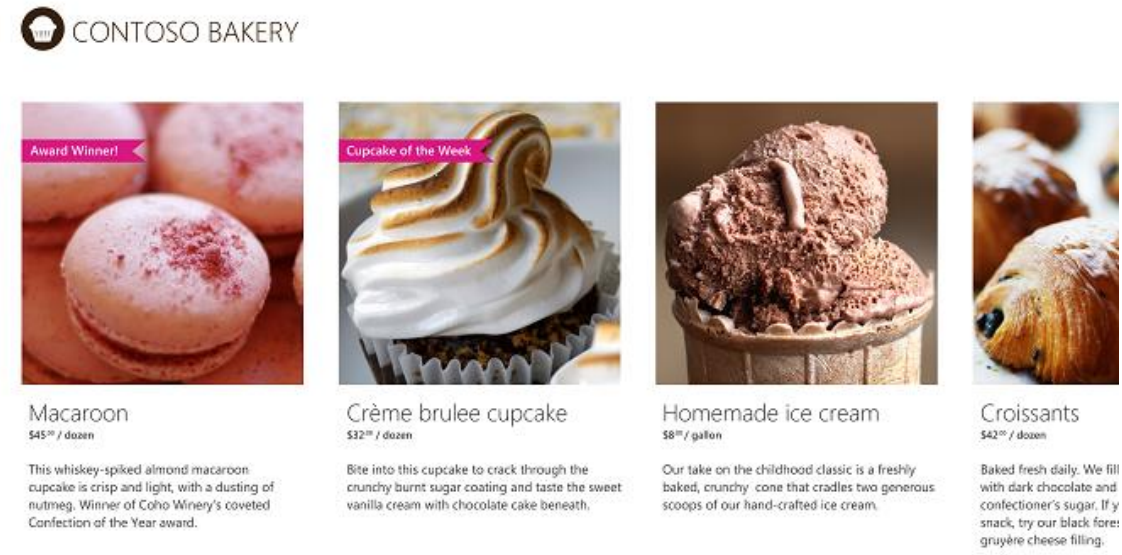
ブランド

Windows ストア アプリにブランド設定を組み込むためにアプリ デザイナーや開発者が利用できる多くのクリエイティブな手段のいくつかを見てみましょう。

ブランドをアプリに組み込む理由

ブランドは会社が認知してほしい品質を定義します。Windows ストア アプリをデザインするときは、ブランドの本質がアプリに組み込まれるようによく考えてデザインを決定する必要があります。ブランドがビジネスを定義するのと同じように、Windows ストア アプリでブランドを表して他のすべてのアプリと区別します。1 つの例を見てみましょう。

これはブランド設定前の Windows ストア アプリの例です。



Contoso Bakery の例は、Microsoft の設計原則に従った場合にアプリのビジュアル（外観）がどのようになるかを示しています。この例は、コンテンツを前面に押し出しているという点では評価できますが、ベーカリーのブランドが効果的に表現されているとは言えません。

次に示すのはブランド設定後の同じアプリの例です。

このバージョンの Contoso Bakery アプリでは、いくつかの明らかな変更によってアプリがいっそう魅力的になっています。店のブランドを組み込むことで、アプリのコンテンツがいっそう説得力のある方法で表現されるようになり、アプリの全体的なフィーリングが店の本質を喚起します。

ブランドを組み込む方法

ブランドの表現は一連のビジュアル要素によって得られます。たとえば、独特なカラー パレット、グラフィックス、レイアウト、写真スタイルがすべて一体となって、ブロードキャスト、印刷、Web、Windows ストア アプリなどのさまざまなメディアで再現することができ、認識してもらえるブランドを形成します。

これらのビジュアル要素は、Windows ストア アプリに固有のビジュアル（外観）を作るためにコードで操作するノブやダイヤルと考えることができます。

ビジュアル要素	説　　明
色	色はブランドを表現するための重要な特性です。ブランドと関連付けられた主要な色を、人々がそのアプリを提供している企業に気付くような方法で適用します。
グラフィックス	グラフィックスを使うと、コンテンツの表現に特徴を追加してブランドを強調することができます。ただし、あまりに多くのグラフィックスを使うと、コンテンツの流れが乱され、ただの飾りのように見えたり、かえって邪魔になる場合があります。
画像	図や写真には、アプリのブランドが反映されていることも必要です。会社で使われている他のコミュニケーションや Web サイトのイメージおよびスタイルを踏襲するようにします。
グリッド	Windows ストア アプリのグリッド システムを使うと、アプリの外見を決めるビジュアル要素に統一感が生まれます。グリッドにより、ブランド設定されたアプリの UI と Windows の他の部分が連携します。

レイアウト	すべてのページのビジュアル要素は、ブランドのコンセプトに沿って配置されている必要があります。また、ページとコンテンツの種類に関係なく常に一貫性を保つ必要があります。
ロゴ	アプリの開発者とブランドをだれもがすぐに認識できるようにロゴを使います。
タイポグラフィ	書体は、Windows ストア アプリの重要な要素です。適切な書体は、色やロゴ、レイアウトと同じぐらい、ブランドにインパクトを与えることができます。使用する書体は、慎重に選ぶようにしてください。

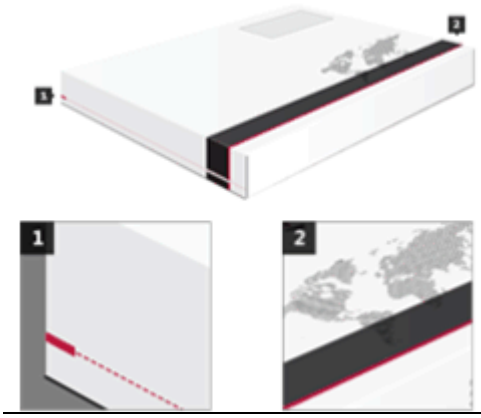
基幹業務アプリの例

基幹業務 (LOB) アプリのブランド設定を行うことは、小売業の場合も企業の場合も重要です。 この例は、宅配便会社用の小売業向けアプリとエンタープライズ アプリに、ブランドがどのように活かされているかを示しています。

会社のブランドの要素

Fabrikam Worldwide Logistics は、3 レベルの宅配サービスを国際的に提供しています。 対象顧客は、配送の準備や管理を容易に行えることを望むスモール ビジネスの事業者です。 顧客へのコア メッセージは、信頼できる国際配送とスムーズなトランザクションです。

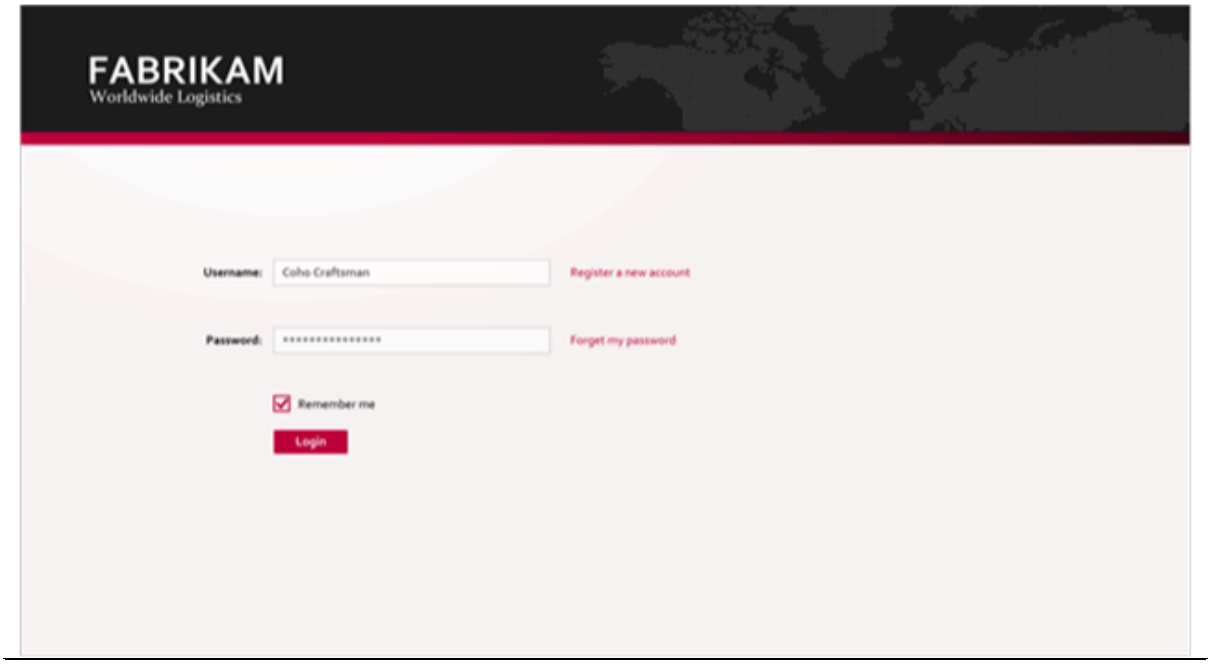
ブランド要素	説明
	メイン ブランドでは、地図や下部のグラデーション ラインなどのグラフィックスにより、信頼性の高いグローバル サービスを提供しようとする情熱を表現しています。
	メイン ブランドに導かれ、各サービスのロゴは、3 レベルのサービスを表現しています。



パッケージに配置された点線や地図などのブランド要素は、アプリ デザインにも使用され、まとまりのあるブランド感を作り出しています。

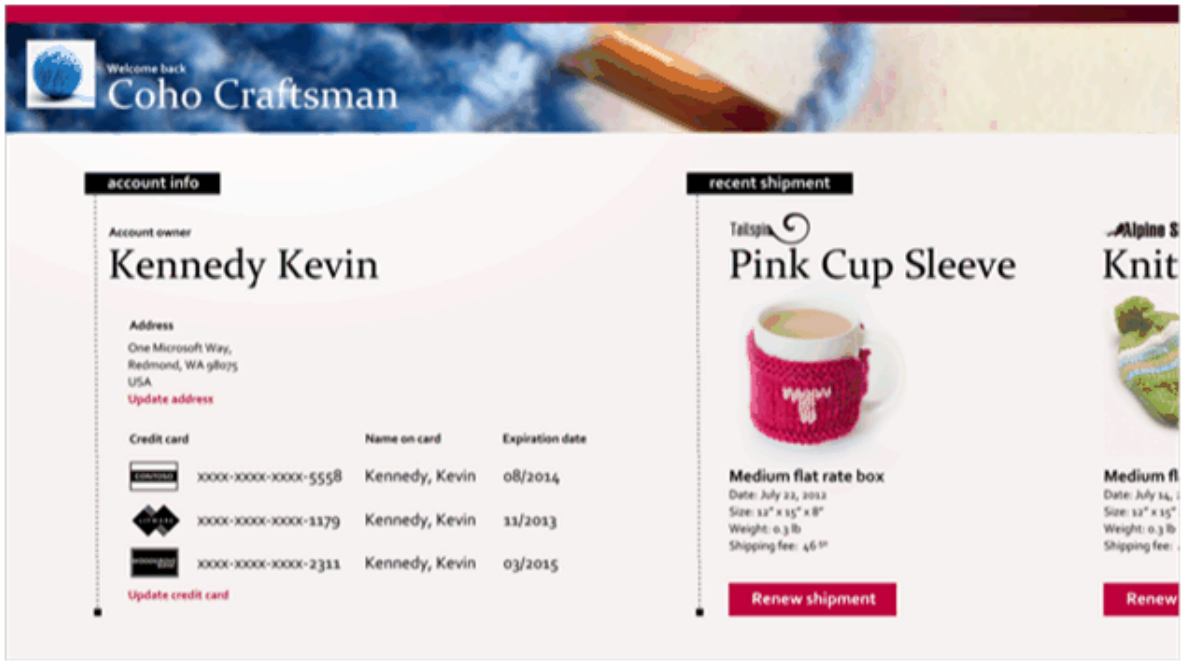
配送の準備

スモール ビジネス顧客は自社アカウントにログオンし、配送を更新します。



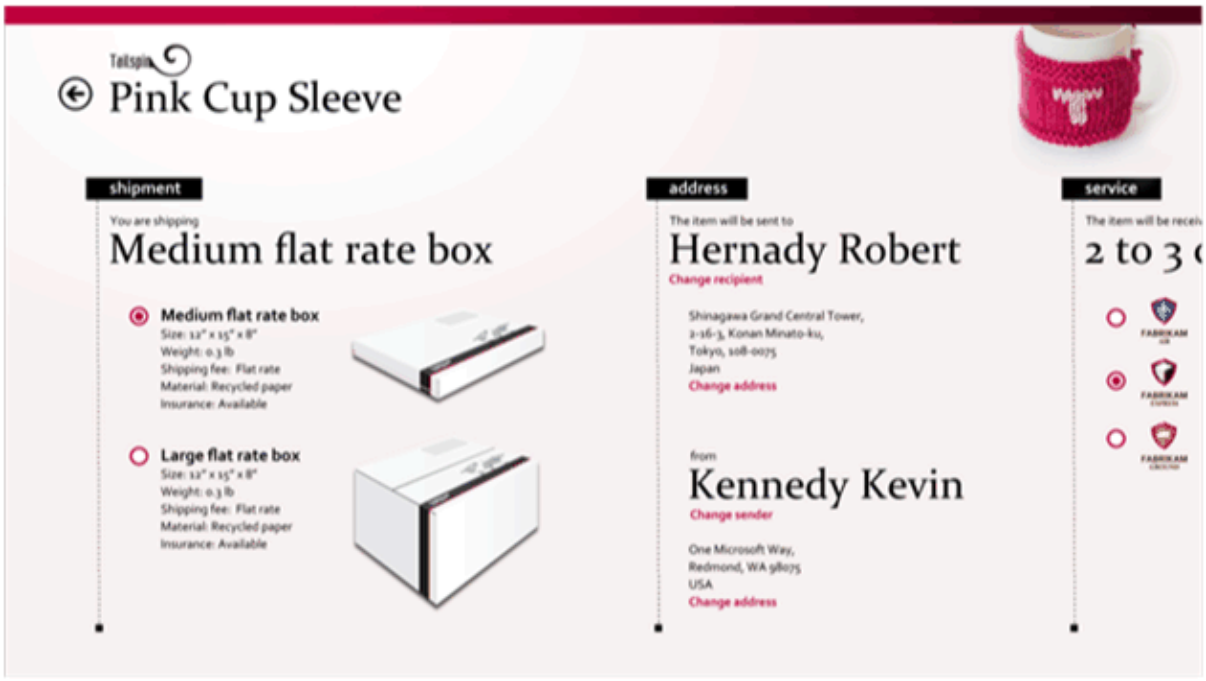
ヘッダーと各ページを通し、ブランド要素として同じ色、グラフィックス、ロゴが使われています。

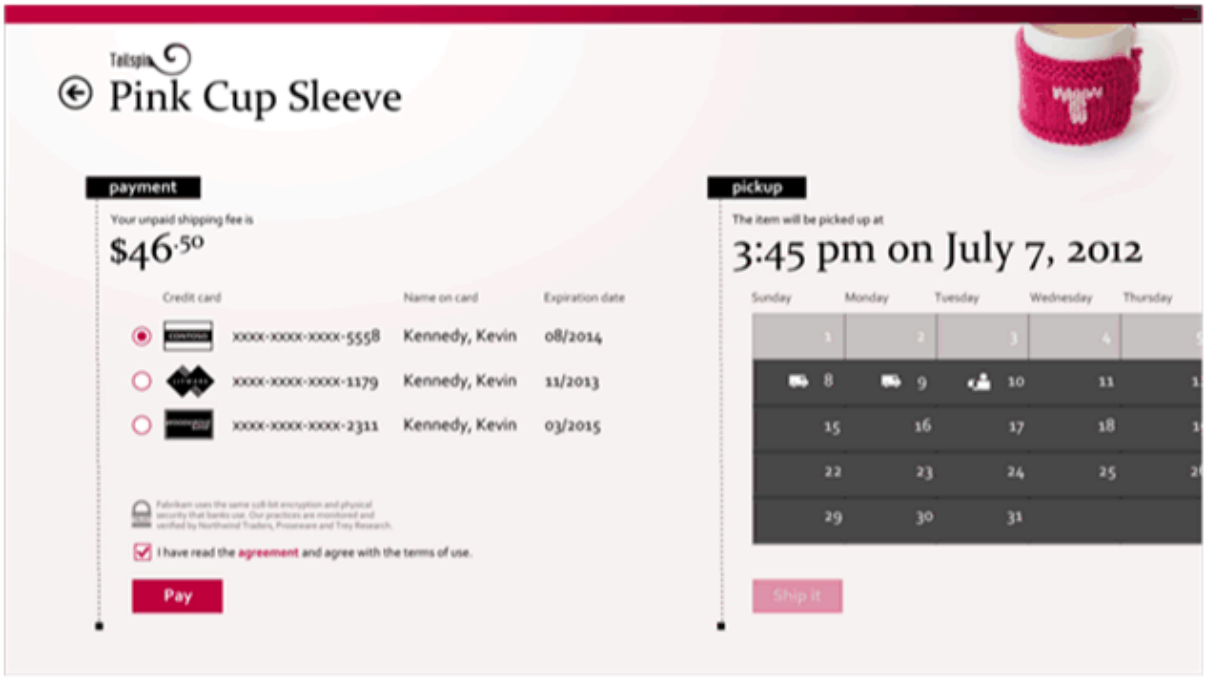
アカウント ページでは、色、グラフィックス (グラデーション ラインと点線)、書体でメイン ブランドを表現しています。顧客名が提携ブランドとしてヘッダーに表示されています。



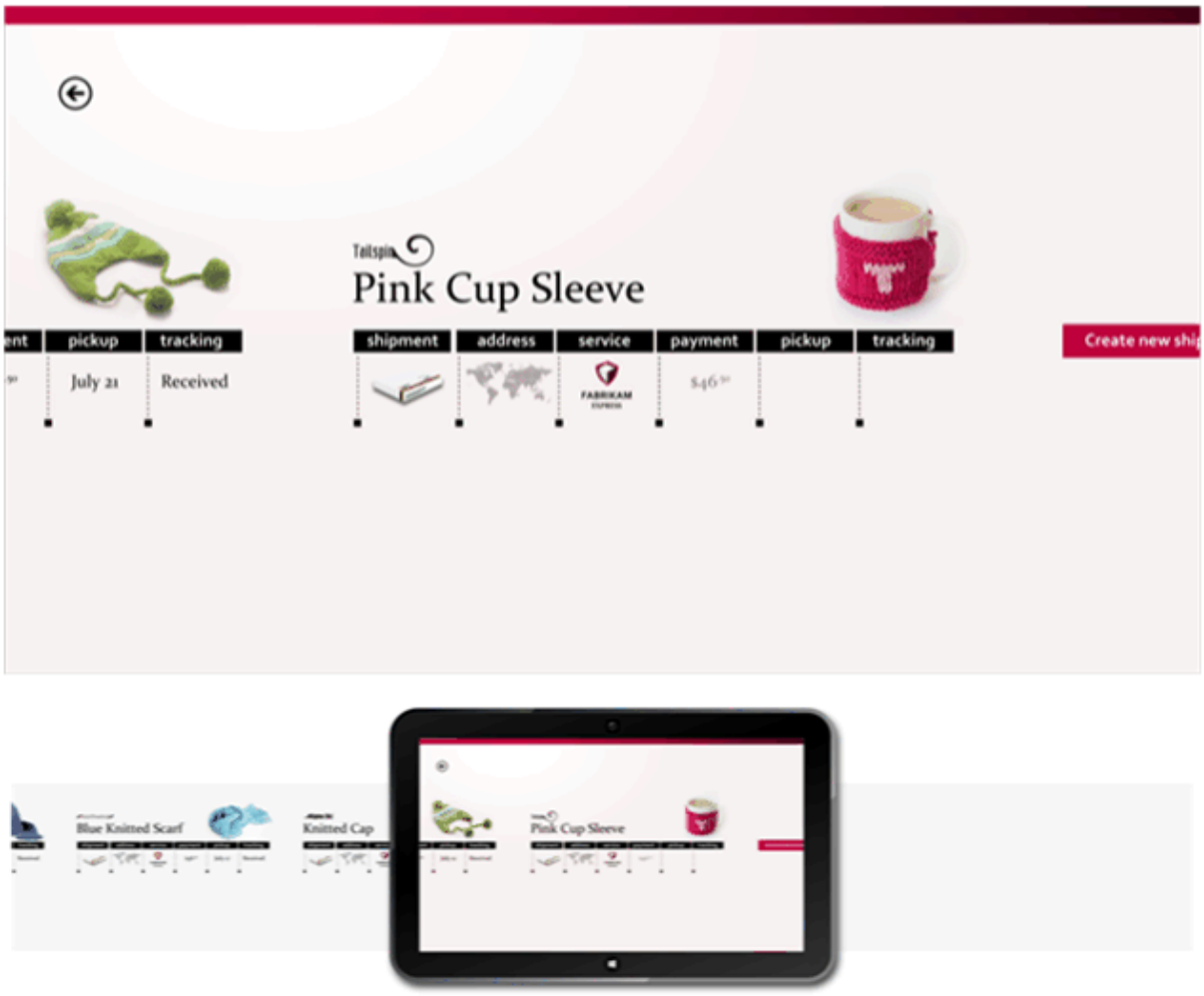
提携ブランドとして表示されている顧客名では、メイン ブランドの書体からインスピレーションを得て、タイトルには Constantia というセリフ フォント、アプリのその他の部分では Corbel というサンセリフ フォントを使っています。

トランザクション ページでは、フラット ナビゲーション パターンを使っています。ユーザーは各セクションをパン表示し、情報に目を通すことも変更することもできます。



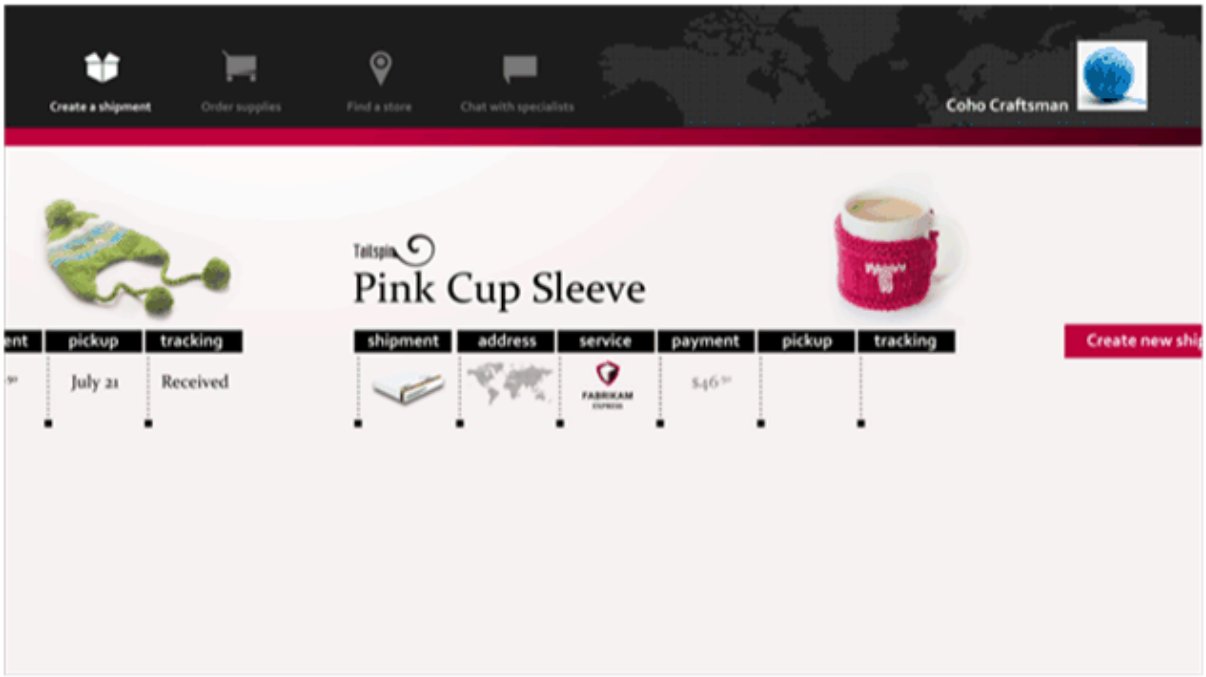


ユーザーはトランザクション ページからセマンティック ズーム ビューにすばやく移動し、すべてのトランザクションを概要情報と共に表示できます。



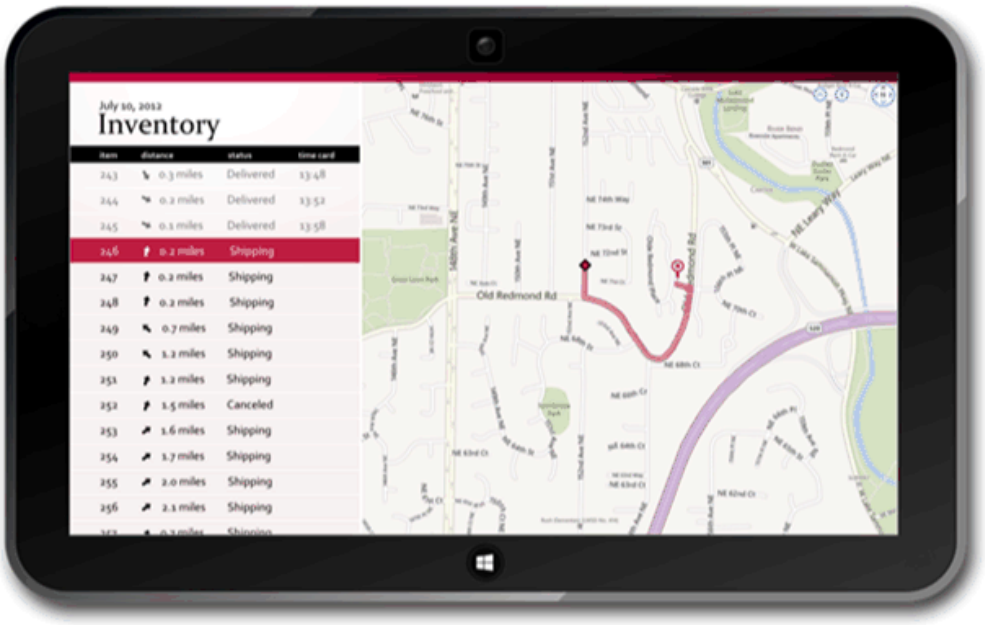
色、グラフィックス、サービス ロゴ、書体などのブランド要素を一貫して使うことで、このビジネスの本質を伝えています。

ユーザーは、ナビゲーション バーを使って他のカテゴリに移動できます。

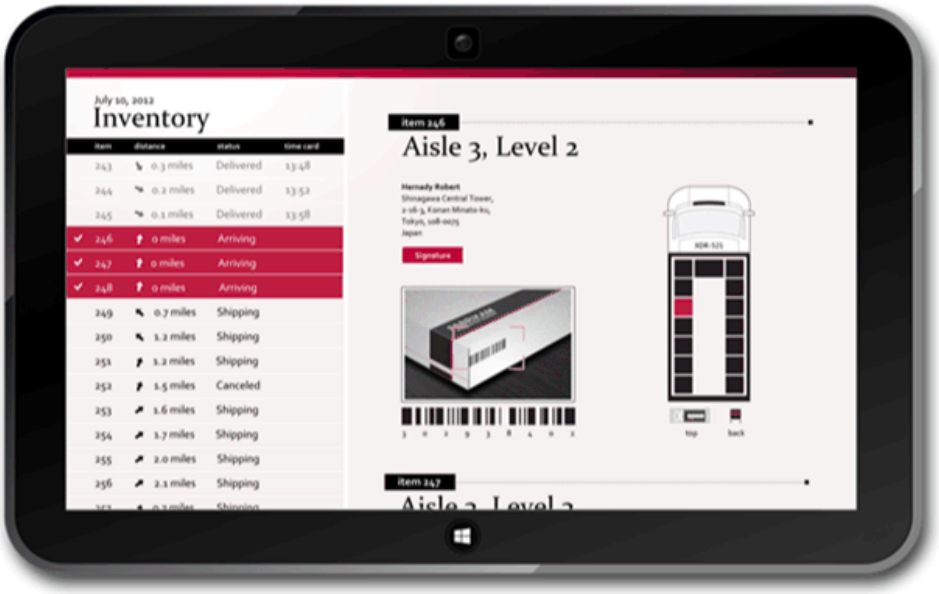


荷物の引き渡し

ドライバーはアプリを使って荷物を顧客に引き渡します。2 列表示の画面を使い、左側にインベントリ、右側に内容を表示しています。ドライバーは配送の詳細情報に容易に移動できます。 グラデーション ラインや共通のコントロールなどのブランド要素が、アプリ全体に一貫して使われています。



ドライバーが目的地に到着すると、右側の列に垂直スクロールの一覧で詳細が表示されます。インベントリ一覧と詳細内容は、同じ方向にスクロールできます。

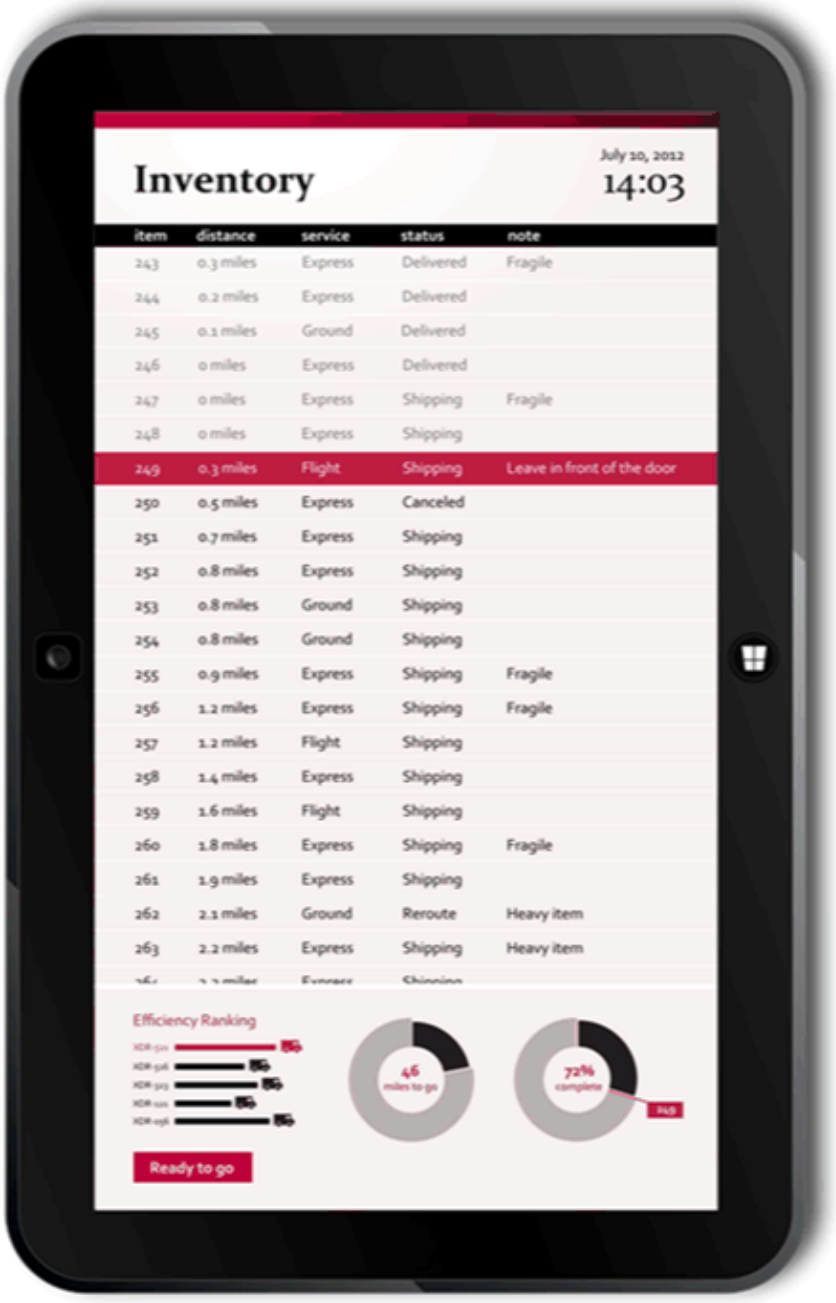


運搬車両内のどこに荷物があるのかをドライバーに示す情報グラフィックスには、アプリの他の部分と同じブランド カラーを活用しています。

ドライバーが顧客に署名を求める際に、アプリでは縦方向レイアウトで署名ページが開き、ブランド設定されたナビゲーション バーが表示されます。ドライバーが顧客にデバイスを渡す際には、縦方向レイアウトの方が便利です。

署名を求めるタイミングは、ブランドを提示する大切な瞬間です。このときユーザーは、エンタープライズ アプリとブランドに触れます。

ドライバーが署名をもらい終えたら、アプリでは残りのアイテムと進捗状況のグラフィックスが表示されます。

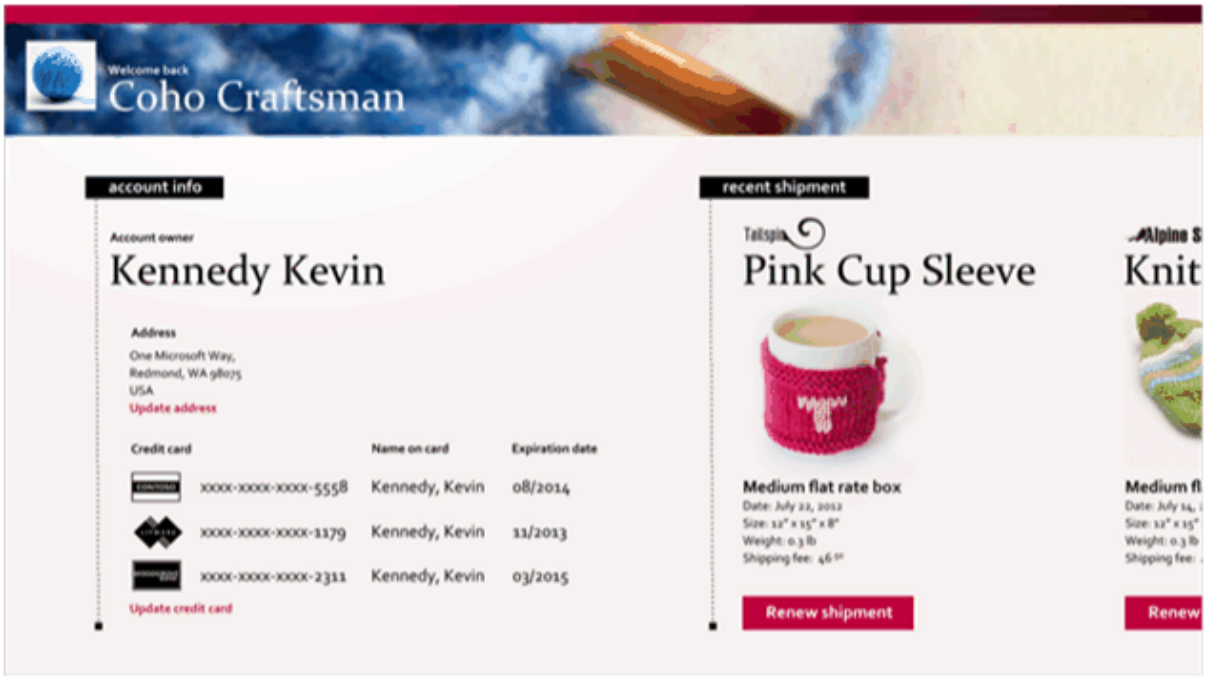


アプリのブランド提携

2 つ以上のブランドどうしの関係をブランド提携と呼びます。複数のブランドを組み入れるアプリを設計する際には、次の点を考慮します。

- どちらのブランドが情報伝達を先導しているか
- 提携先のブランドは、情報伝達においてどのような役割を果たしているか

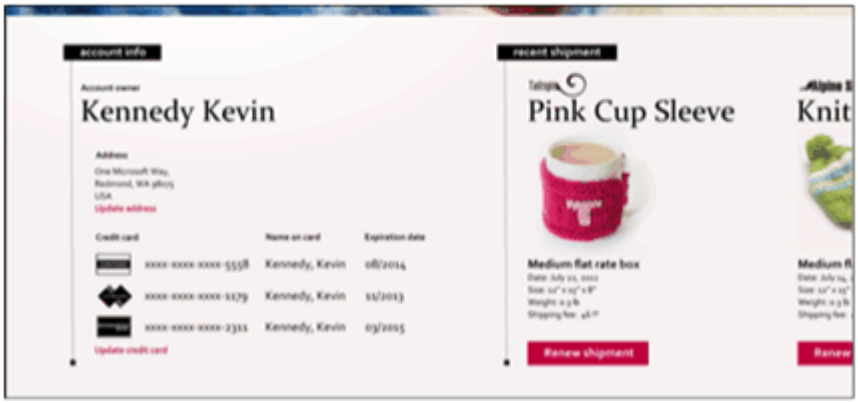
これらの点をきちんと考慮していない Windows ストア アプリには、顧客を混乱させ、ブランド力を弱めるリスクがあります。



Fabrikam の例では、ビジュアル要素を一貫して全面に出してブランドを強化することにより、ブランド提携に対処しています。たとえば、Fabrikam のブランド カラー、グラフィックス、レイアウト、書体がアプリ全体に使用され、限られたスタイル バリエーションを守っています。アプリの各ページでは、アプリが Fabrikam によって配信されていることを明確に伝えています。

メイン ブランドに対して Coho Craftsman ブランドがメインではなく提携先であることを表すために、Coho Craftsman のロゴ、書体、イメージは Fabrikam のレイアウトに挿入された

形になっています。たとえば、Fabrikam の別のスモール ビジネス パートナーが、現在の Coho Craftsman の位置にどのように適合するか、考えてみることができます。



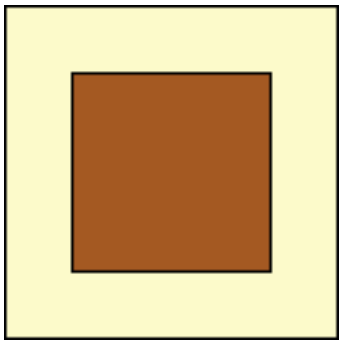
この例には他のブランドも登場しています。ページの左側のクレジット カード会社は、サイズの小さい表示、コンテンツ固有の配置、限定された色使いなどにより、下位レベルのブランドとして表示されています。ページの右側には、衣料製品メーカーのブランドが縮小サイズの単色ロゴで表示されており、メイン ブランドのデザイン スタイルに適合しています。すべての場合において、提携先ブランドは鮮明なままですが、メイン ブランドほど目立たないようにビジュアル要素が修正されています。

固有のブランドを喚起する例

各々のビジュアル要素で異なる感情や感覚を喚起する方法の例をいくつか示します。各々のビジュアル要素について、レイアウトの作成方法の具体的な詳細とその重要性を説明します。

Contoso Downtown Café

Contoso Downtown Café は正真正銘の品質で知られたブランドです。手作りの料理と自家醸造のスペシャル ドリンクが自慢です。この例では、色、グリッド、レイアウト、タイポグラフィがブランドの喚起に主として使われているビジュアル要素です。



色　暖かいダーク グレー、オフホワイト、銅色がこのデザインにリッチ感を与えています。暖色のパレットは、手作りのパンの耳、スープ、ペストリーで目にするのと同じ色を喚起します。

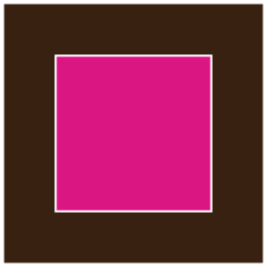
グリッド　この例のコンテンツでは、意図的にグリッドを "崩して" あります。このブランドではメニュー コンテンツが重要であるため、上部の余白を減らしてメニューのスペースを大きくしています。会社名、紹介文、住所が中央に寄るように左側の余白を調整してあり、コンテンツに余裕を与えています。これらの調整はすべて一貫して、このアプリの他のすべてのページにも適用されています。

レイアウト　客がこの店のリピーターになる理由はメニューにあるため、デザインは意図的にメニューらしくしてあります。サンドイッチや他のメニュー品目のように、レイアウトは手作り感を増すように意図されています。オフホワイトの背景色は紙を思い出させ、トーンは勧誘的であり、コンテンツは整然としていて、縦の区切り線と強い左揃えのテキストを使っています。

タイポグラフィ　このデザインで主に使われている書体は Copperplate Gothic Bold であり、手彫りの木の看板に似た独特の外観を会社名に与えています。Script MT Bold と Segoe UI は付随する書体です。Script フォントはメニューの見出しに控えめに使われ、手書きスタイルをもたらしています。Segoe UI は、メニューを選びやすいように本文のテキストとして使われています。

Contoso French Bakery ブランド

Contoso French Bakery は気ままな店として知られたブランドです。甘いものへの欲望を満たすために、多くの客が訪れます。 この例では、色、画像、レイアウトが、ブランドの連想に使われているビジュアル要素です。

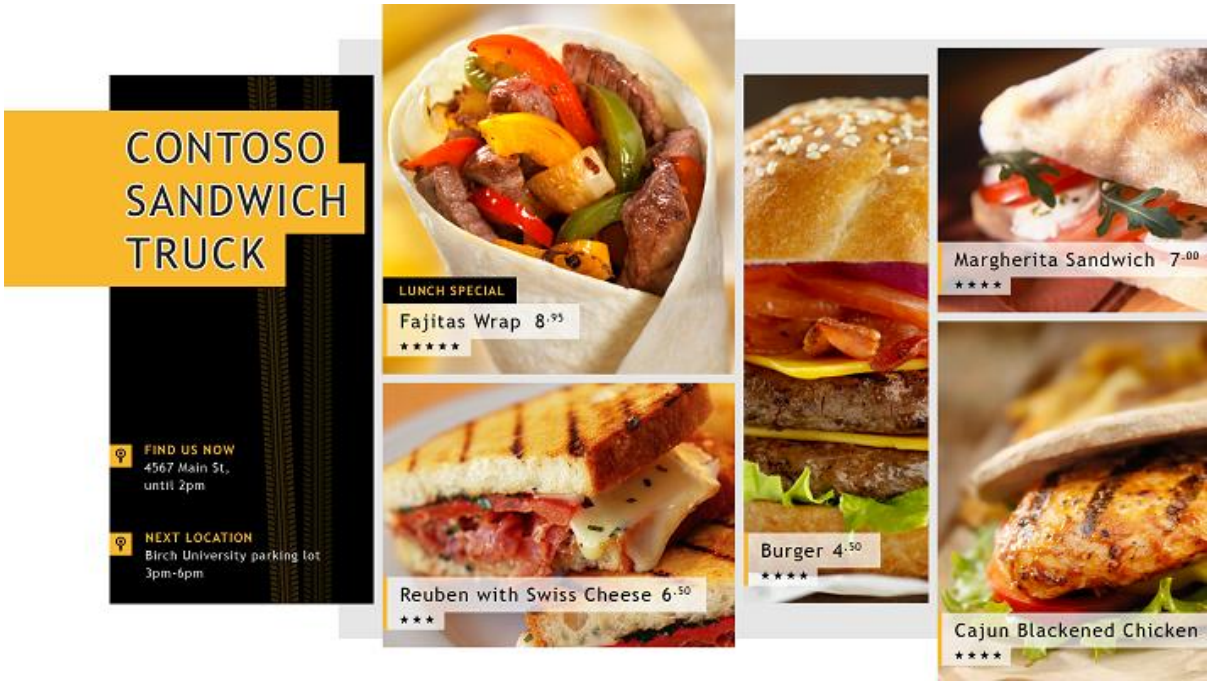


色　客にベーカリーを連想させるには茶とピンクの 2 色だけのカラー パレットで十分です。茶はチョコレートを連想させる主要な色であり、ピンクは砂糖をまぶしたクッキーやケーキを喚起するアクセント カラーです。どちらの色も、それに関連する風合いだけでなく、ブランドにとっても意味があります。茶はフルカラーの写真を目立たせるリッチな背景として十分な中間色であり、ピンクはコンテンツの特定の部分を強調するのに十分な明るさです。

画像　写真はこのデザインの焦点です。食欲をそそる画像にするために時間と専門知識が費やされたことは明らかです。その結果、ベーカリーの商品の画像でこのブランドが明確になっています。また、コンテンツのカテゴリを定義するためにも画像が使われています。

レイアウト　この例には、標準的な Windows ストア アプリのデザインとは異なる感じがあります。最も大きな違いは、正方形のタイルがないことです。タイルとグリッドは存在しますが、画像はベーカリーにありそうなもの (カップケーキの容器の円形や波形の形状) に似せてくり抜かれています。

Contoso Sandwich Truck ブランド



Contoso Sandwich Truck は都会的、社会的、場所との結び付きで知られたブランドです。このブランドは、腕のいいシェフのクルーと一団の車両によって支えられ、活動的な人々を対象としています。この例では、グラフィックス、グリッド、ロゴがブランドの喚起に使われているビジュアル要素です。

グラフィックス　メニューの選択肢が中心になっており、雑然とするほど過剰にグラフィックスは使われていません。ただし、グラフィックスがこのブランドを確立する役割を担っています。たとえば、"星印" は顧客の評価システムを暗示し、色と形状は道路標識を思わせるものになっています。また、ザラザラしたアスファルト道路のグラフィックスを取り入れた濃い色のボックスを背景に、場所ベースの情報が配置されています。

	**グリッド** 標準のグリッドがこのデザインのすべての部分をアンカーしており、都市の道路地図を見下ろしたような構造を作り出しています。この規則的な表現に対して、完全に断ち切られて重ねられたコンテンツとグラフィックスがこの例を引き立てています。コンテンツの全体的な表現は、忙しくて短時間で決断する必要のある人にアピールする手段として、目立つ直接的なものになっています。
	**ロゴ** この例では会社のロゴは使われていません。この例ではロゴの位置に会社名がテキストで示されており、全体構成に埋め込まれた会社名にまず目がいきます。ロゴつまり名前は左上に単純に配置されてはいません。
	**色** アクセント カラーとして黒と黄が使われています。個人的な感覚では、黒と黄は道路標識や路面表示などのように都会で見られる色でもあります。

アプリを複数のページでブランドを印象付ける例

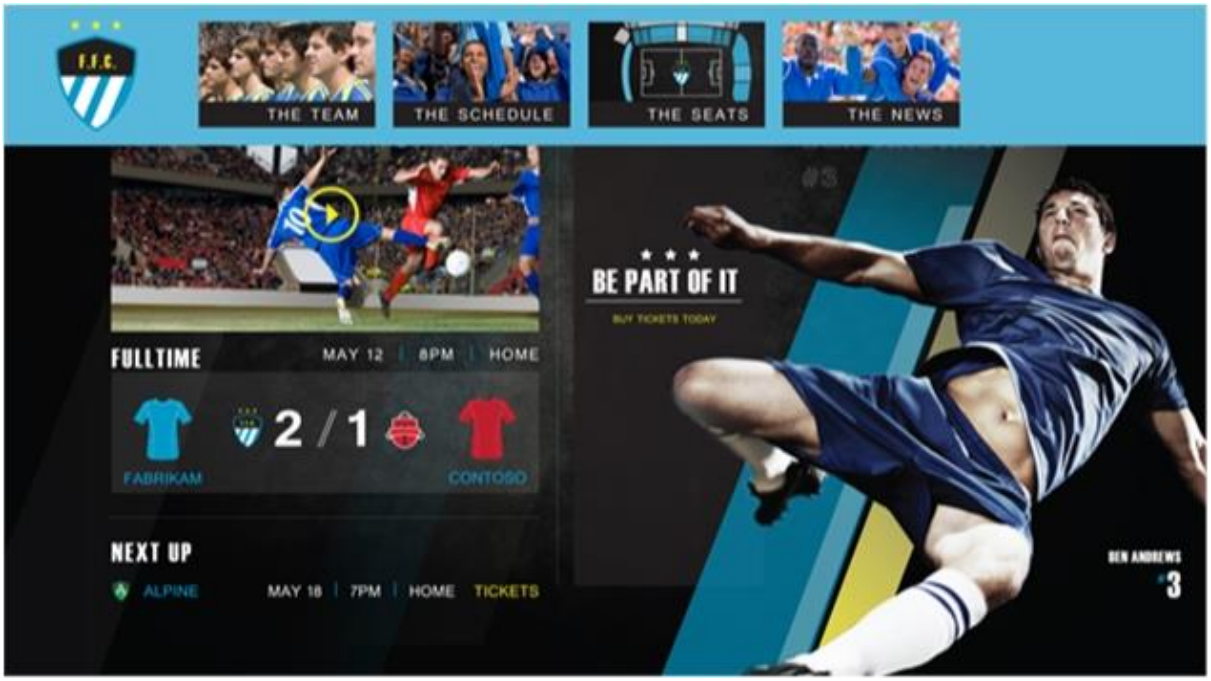
アプリで 2 番目に表示されるコンテンツが豊富なページでブランドを印象付けることは、ランディング ページでのブランドの扱いと同様に重要です。この例では、架空のスポーツ チームのアプリを使って、アプリの複数のページにブランドを組み込む方法を説明します。

プロ サッカー チーム Fabrikam FC

プロ サッカー チーム Fabrikam FC のファンは、応援しているチームのお宝映像や、お気に入りの選手の詳しい情報、チームの最新情報を、どこからでも入手できるようにすることを希望しています。ランディング ページのデザインには、このブランドの大胆でダイナミックな性質が反映されています。

色、タイポグラフィ、グラフィックスによって、このブランドの大胆でダイナミックな性質を強調しています。

このアプリでは、フラット ナビゲーション パターンを採用しています。ユーザーは上部のナビゲーション バーをスワイプして、アプリの他のビューに移動します。



ナビゲーション アプリ バーにも、ブランド要素として同じ色とロゴが使われています。黒い背景に明るいチーム カラーを配すことで強いコントラストを作り、チームの大胆さを表現しています。また、対話操作ができる部分はアプリ全体で黄色に統一されています。

ナビゲーション バーを使って、チームの選手を表示できます。

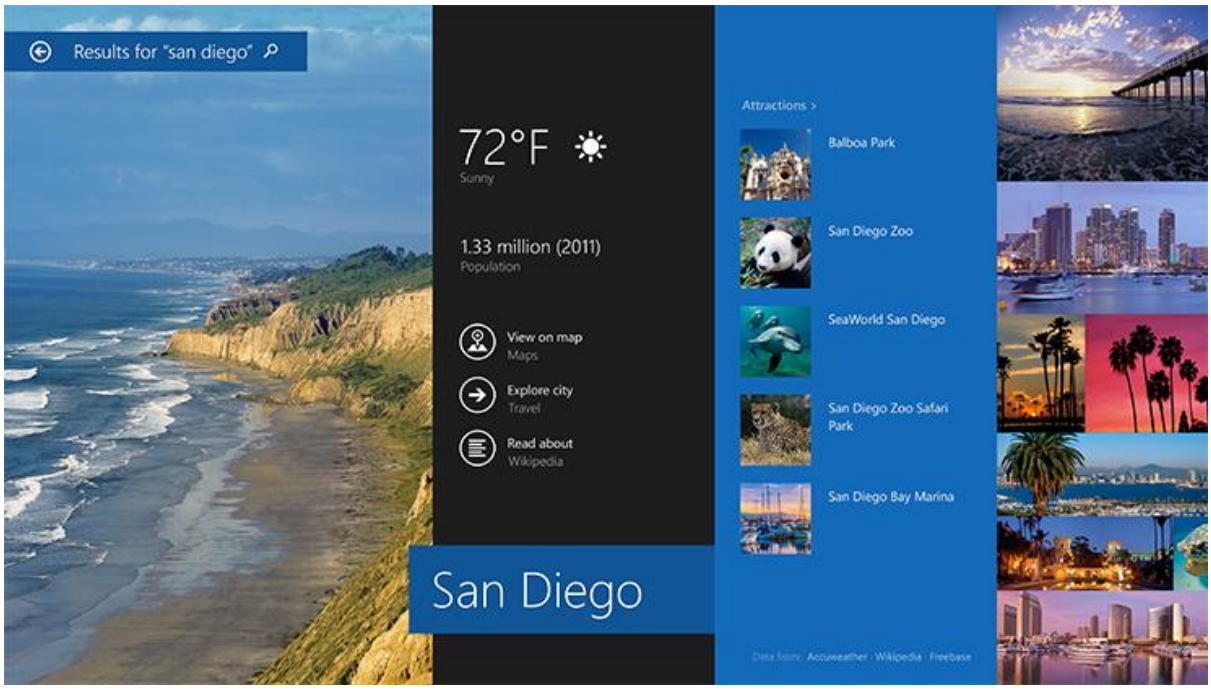
フォントの種類は、アプリのさまざまな表示部分で一貫しています。sans-serif 書体によって、力強さと大胆さというチームの特徴を表現しています。すべて大文字のフォント スタイルは、特定のテキスト ラベルでグラフィック要素のような印象を感じられるようにするために使われています。

特定の選手にすばやく移動できるように、縮小表示すると、このページのセマンティック ズーム ビューが表示されるようになっています。

選手が動いている写真はグリッドをはみ出し、エネルギーに満ちたダイナミックなレイアウトになっています。これはチーム ページとランディング ページで使われます。

シンプルなアイコンとグラフィックスを使うことで、コンテンツの親しみやすさとアクセス性を高め、アプリ全体の対話性を向上し、ユーザーの探求心を刺激することができます。

レイアウト



コンテンツおよびページ間のナビゲーションの両方について、アプリのレイアウトはアプリの全体的なデザインおよび操作感にとって非常に重要です。レイアウトによって階層を作り、コンテンツを整理し、ブランドを反映させますアプリのユーザー エクスペリエンスは、ページとコンテンツの構造に依存します。

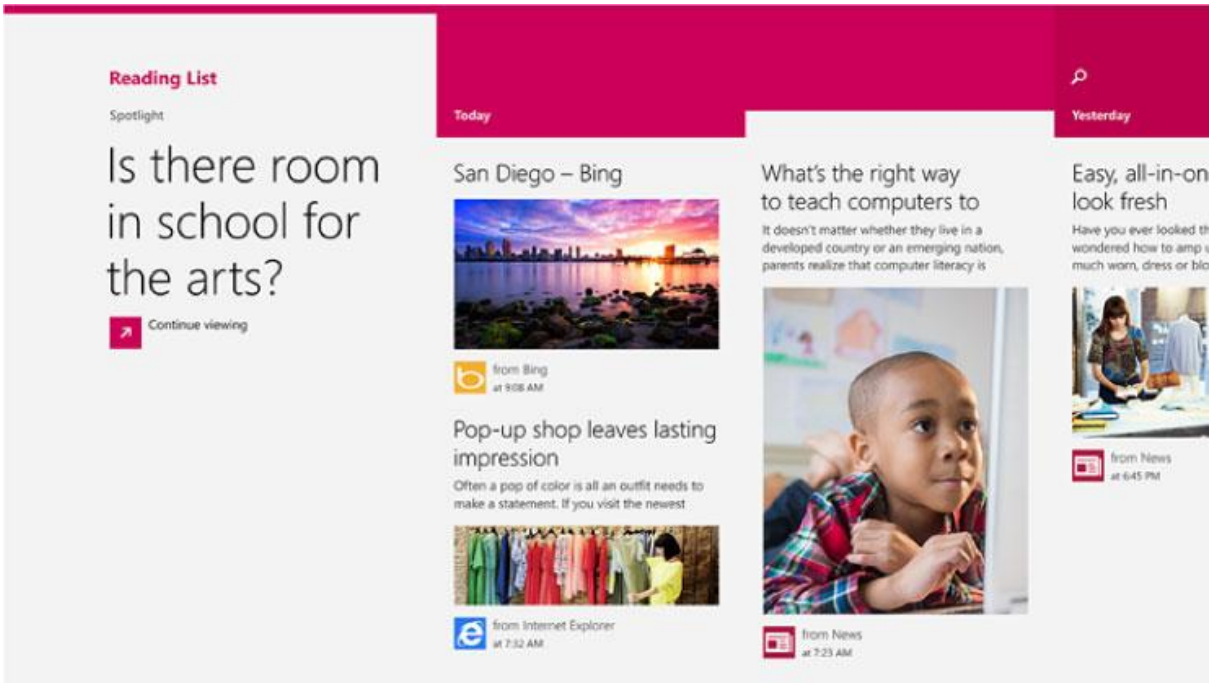
Windows におけるレイアウト

Windows のレイアウトは真にデジタルです。デジタル メディアを取り入れ、多くの場合にデジタル世界で不必要であるか誤解を招きやすい実際のメタファーは避けます。制約のない自由なデジタル レイアウトは、ユーザーにとって強力で美しいアプリとエクスペリエンスを実現します。

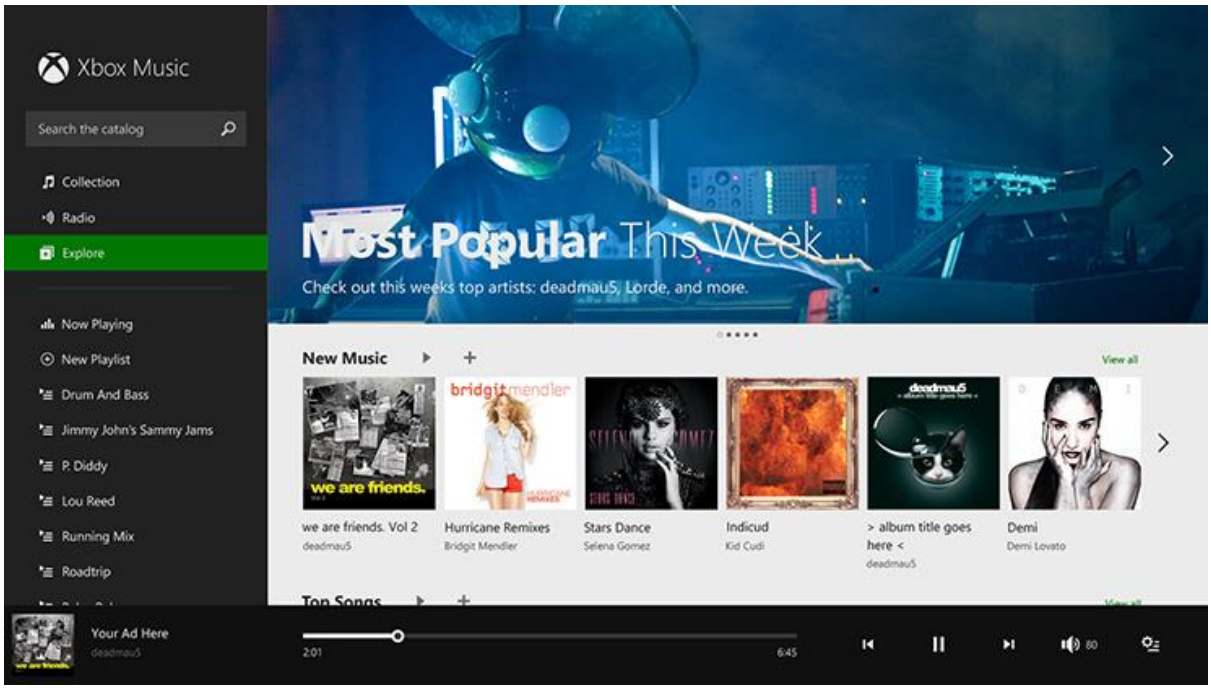
Windows のレイアウトは、明確ですっきりとした印象を持ち、キャンバス上の視覚的な混乱を抑制することを試みています。アプリのエッジのアプリ バーにコントロールを移動することもできます。これによって、アプリの重要な機能をキャンバスの目立つ場所に配置して、アプリの操作性を向上させることができます。

アプリのコンテンツに応じて、大きな対話領域を含む明るく軽快な構成や、多彩なコンテンツとコマンドを確認できる高密度なエクスペリエンスを作成できます。たとえば、リーディ

ング リスト アプリは、オープンで最小限のレイアウトを持ちます。アプリで何を表示し、キャンバスから何を取り除くか、そしてレイアウトの選択がアプリのエクスペリエンスと使用にどのように影響するかについて考慮してください。

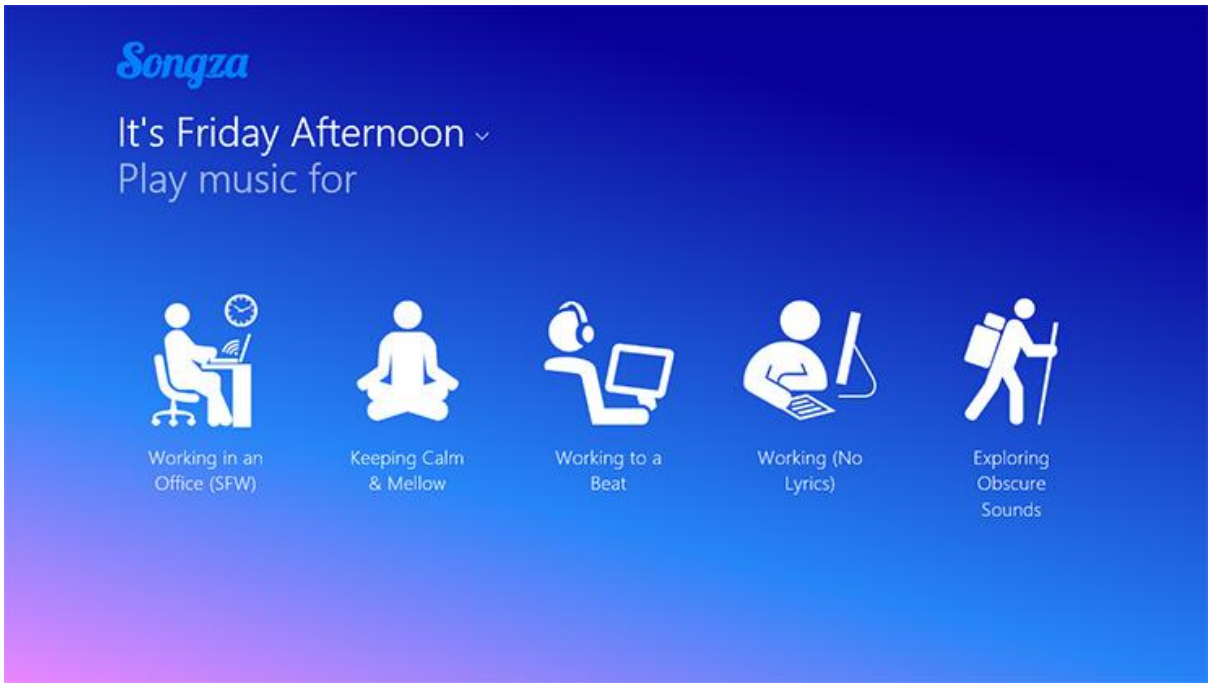


Xbox Music アプリは、広いコンテンツ領域を同時に表示するために、非伝統的な縦向きのグリッドと縦向きのパンを使います。一番下の永続的なアプリ バーでは、ユーザーがアプリの他の領域を確認しながら再生リスト内を迅速に移動できるように、よく使われるコントロールが常に提示されます。



レイアウトの技術革新

ユーザーのシナリオに基づいて変化する音楽アプリである Songza は、最小限の横長のレイアウトを使います。このアプリを使う大多数のユーザーは、ナビゲーションではなく、音楽を聴くことが主であるために、シンプルであることが実際にはより効果的です。



多くのアプリでは、ナビゲーションの階層システムが使われます。ハブ (ランディング ページ) は、ユーザーにとってのアプリの入り口であり、さまざまなレベルのコンテンツを開くための起点となります。パンできるビューに豊富なコンテンツが表示され、新しい情報や興味深い項目がひとめでわかり、そこからさらにアプリのコンテンツを探すことができます。ハブ コントロールは、ブランドに合わせてアプリをカスタマイズする柔軟性とフレームワークを提供します。

階層型ナビゲーションは、アプリ レイアウトの単なる 1 つのモデルであり、アプリではブランドに最適なレイアウトとナビゲーション モデルを利用し、アプリ ユーザーに最高のエクスペリエンスを提供します。構成とレイアウトには創造性を発揮でき、作成者がユニークな何かを創り出す大きな機会が提供されます。

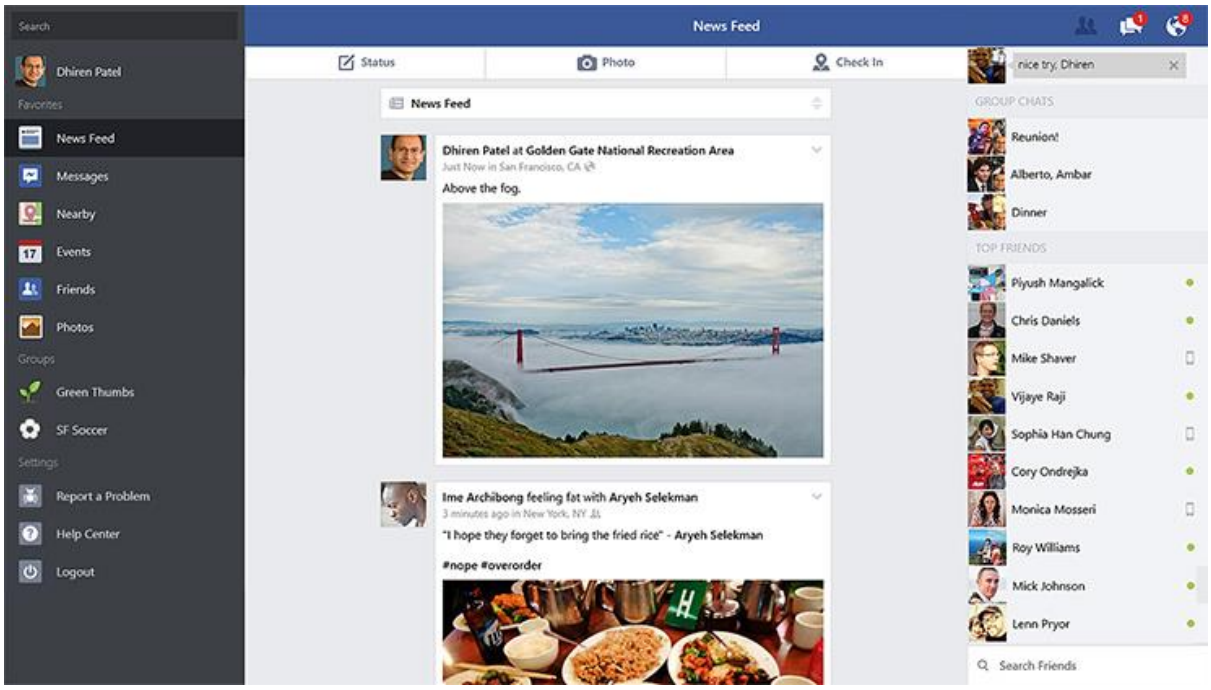
アプリの向きとして縦向きと横向きの両方を考慮します。コンテンツを配置するグリッドが縦向きまたは横向きのいずれの形式で最も効果的であるかを特定します。Windows の最新のデザイン言語を利用するアプリが増えるに伴って、アプリのレイアウトとデザインに対す

るアプローチが各アプリの個別のニーズとブランドに合わせてさらに進化します。レイアウトは、イノベーションを創出し、アプリの卓越性を示す最適な部分です。

例

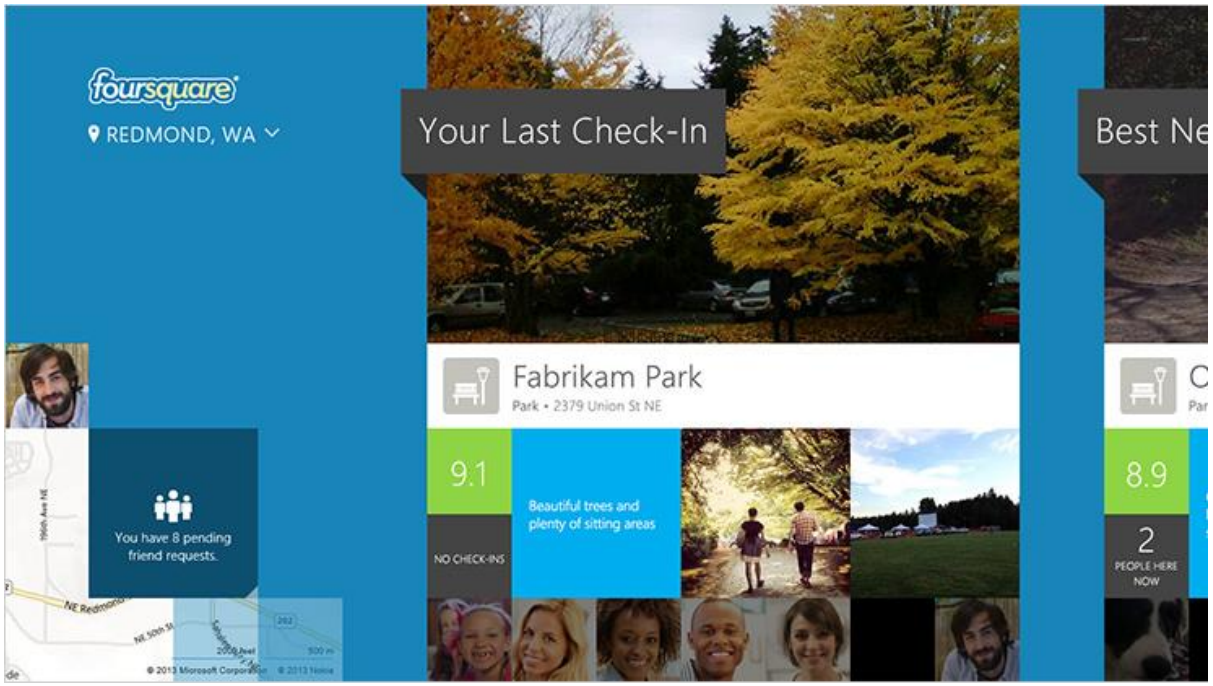
Facebook

Facebook では、キャンバス上で多数のコンテンツとコマンドを整理するために強力なグリッド ベースのレイアウトを使い、Facebook が提供するすべてのサービスをユーザーが直ちに利用できるようにしています。

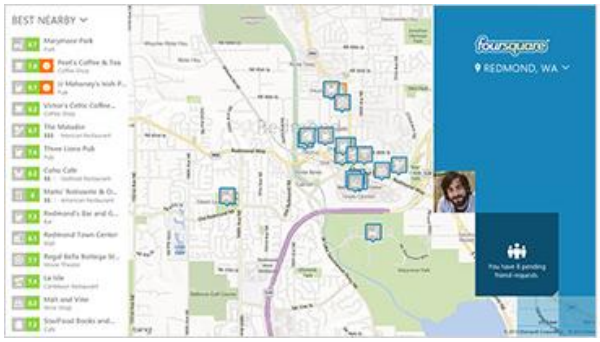


Foursquare

Foursquare は、場所に関するアプリであり、新しいエクスペリエンスを提供します。アプリのレイアウトは、注目される各場所を示す目立つイメージによりこの用途を反映します。イメージの下には評価と確認が示され、ユーザーがひとめで場所を評価し、タップによりさらに詳細を確認できる統一された構成が得られます。

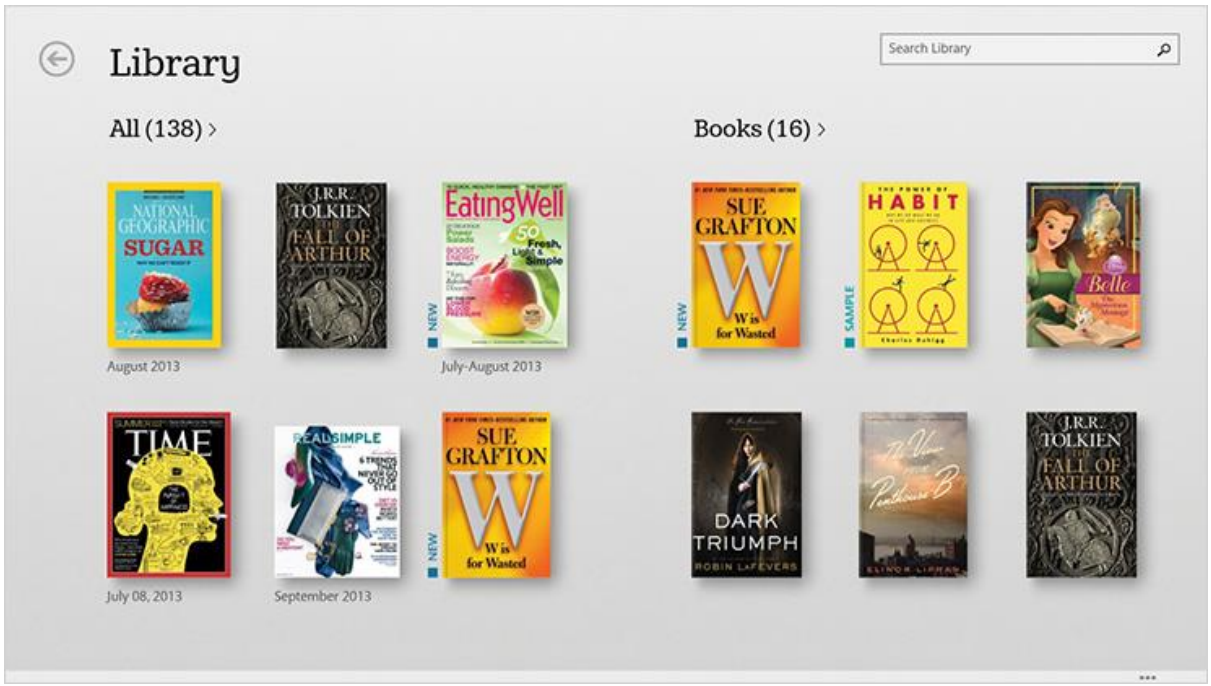


Foursquare のレイアウトは動的であり、アプリの各領域でコンテンツに最も適合するように自身で再構築されます。アプリの最も左側にパンすると、近隣の人気のあるスポットの地図が生成されます。場所をタップすると、すべてのチェックインとユーザーからのコメントを示す詳細なページが表示されます。こうした表示のいずれかをタップすると、推奨のスニペットが表示されます。

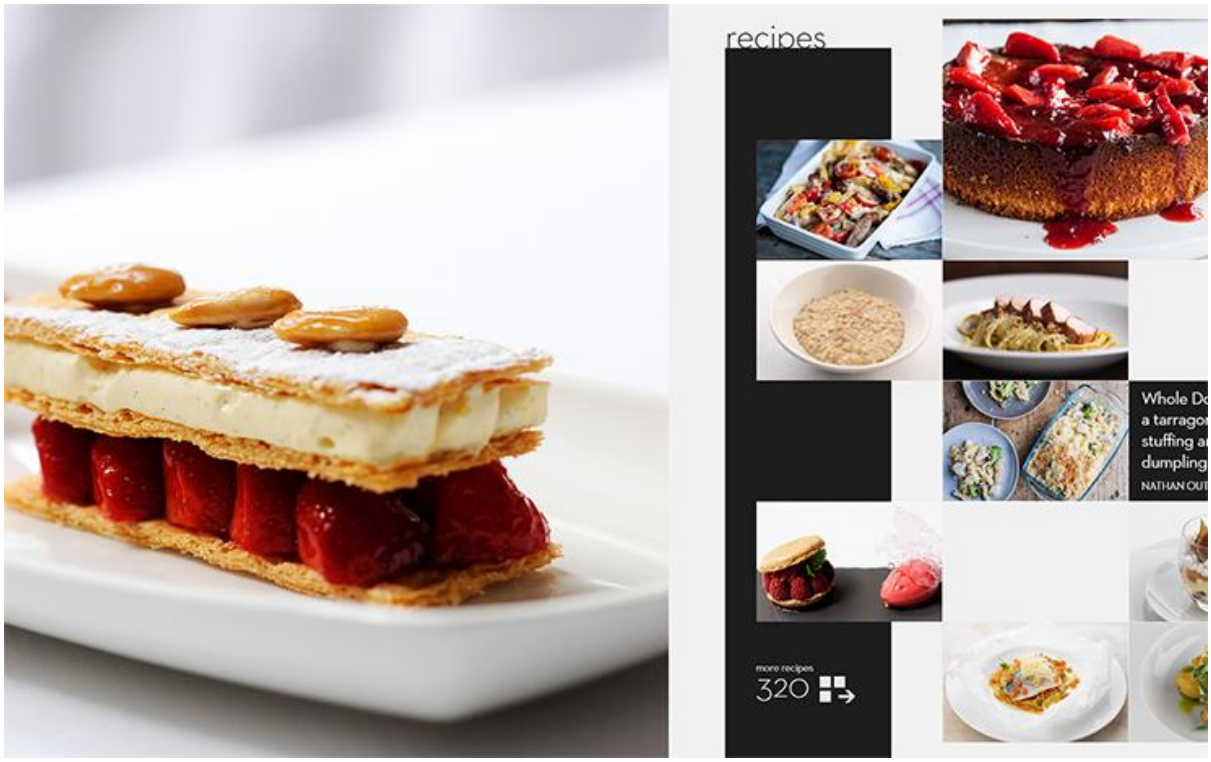


NOOK

Barnes and Noble による NOOK アプリは、基本概念として階層型のハブ ベースのモデルを使いますが、差別化された NOOK ブランドのエクスペリエンスをもたらすために緊密にカスタマイズしています。単なるカバー イメージを表示するのではなく、各書籍の下のかすかな影により現実的な書籍のニュアンスを示しています。この場合、現実のアナログ性を示すことで、ユーザーのエクスペリエンスが拡張します。



アニメーション



アニメーションを使うことで、美しさ、力強さ、個性をアプリに与えることができます。アニメーションは、状態の切り替えを滑らかな動きとして表現することで、ユーザーが現在の状態を認識しやすくします。"軽快かつ柔軟" という原則に従い、効果的に演出されたアニメーションを作成してください。アプリはユーザーの入力に対してスムーズに、連続的に応答する必要があります。

Windows のアニメーション

Windows は完全なデジタル世界であるため、現実世界の物体のような動きをする必要がなく、またそのような動きを模倣する必要もありません。Windows では、動きのぼかしや、誇張された跳ね返りの動きや伸びる動きなどの誇張した振る舞いをしないようになっています。Windows のアニメーションはスムーズで目的に即しており、オブジェクトは引っかかることなく容易に動きます。

パンとスクロールだけでもコンテンツを閲覧することはできますが、セマンティック ズームを使うと、高度なナビゲーション機能や整理機能を利用できるようになります。セマンティック ズームを使うと、大量のコンテンツをナビゲートするときに距離が長いと感じるこ

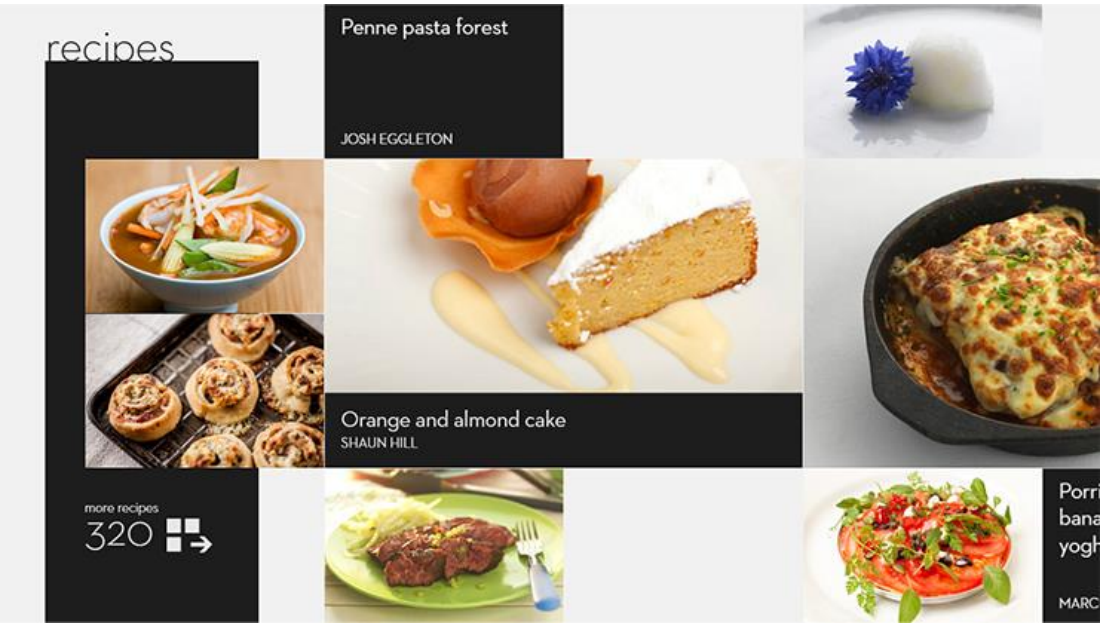
とがはるかに少なくなり、コンテンツ内の目的とする位置にすばやく簡単にアクセスできます。

アニメーションの使い方

静的なレイアウトにビジュアルな階層があるように、アニメーションにもビジュアルな階層が必要です。レイアウトのすべての要素に重要な役割があるわけではありません。最も重要な要素を特定して、必要に応じてアニメーションを適用してください。

Great British Chefs は美しいページ切り替えアニメーションを備え、生き生きとした印象を与えます。このアプリではセマンティック ズームによって容易なナビゲーションを実現しており、ナビゲーション バーの欠点をうまく補っています。

レシピ セクションでは、特にユーザーの興味を惹き付ける必要があります。ホバーするとタイルが裏返しになり、料理の名前とシェフの名前が表示されます。この驚くような要素は、ユーザーを喜ばせ、このアプリの豊富なコンテンツをもっと見たいという欲求を与えます。



アニメーションを使って、アプリの特徴とブランドをさらに明確に打ち出すこともできます。Netflix アプリのスプラッシュ スクリーンのアニメーションは控えめでありながら美しく、興味を惹き付けなくても効果的なアニメーションが作成できることがわかります。アプリの読み込みが始まると、赤色の背景が暗くなってロゴが中央画面に浮かび上がり、進行状況を示す円が表示されます。

Windows Phone 用のアニメーション、モーション、出力

Windows Phone アプリには、アプリ内で触れたときの生きているような感覚を実現するため、モーション、サウンド、バイブレーションを使ってください。このような効果や出力は、タッチを増強し、ユーザーが操作や進行中のタスクのフィードバックを得られるようにするのに役立ちます。

モーションのデザイン

モーションは、単にものを動かすだけではありません。Windows Phone では、モーションはユーザーがアプリに入り込み、タッチを通じて操作するための物理的なエコシステムを作るツールです。エクスペリエンスの品質は、ユーザーに対するアプリの反応の良さと、UI が伝える個性の種類によって変わります。

Windows Phone には、2 種類のモーションがあります。

- 切り替え: 切り替えは、ユーザーの操作によってトリガーされ、UI を移動するときに感じることができる階層リンクをユーザーに提供します。ユーザーをガイドし、今いる場所を伝えるため、さまざまな切り替えが作られています。アプリ間には大きな切り替えが使われますが、アプリ内の移動には小さい切り替えが使われます。
- アニメーション: ビュー内のローカル要素に制限された視覚的なフィードバックですが、常にユーザーの操作の結果として表示されるわけではありません。1 つの例は、デバイスのロックを解除する方法をユーザーに伝えるロック画面の "ホップ" モーションです。

モーションがアプリで目的を果たしていることを確認します。優れた Windows Phone アプリは、モーションを使って UI を生き生きさせています。モーションは次の条件を満たす必要があります。

- タップしなくても情報を表示する。
- ユーザーの動作に基づいてフィードバックを提供する。
- タッチ ターゲットを操作する方法をユーザーに伝える。
- 前または次のビューに移動する方法を示す。

モーションの 1 つの簡単な例は、Windows Phone の編集ボックスの動作にあります。編集ボックスは、拡大するときスナップされるだけではありません。そのサイズ変更がアニメーション化されます。

モーションの効果

ユーザーがアプリ内で費やす時間が増えたり、アプリのタスクが高度になると、高品質なモーションの重要性が増します。ユーザーが認知的負荷とアプリの使いやすさを感じる方法を変えるために使うことができます。モーションには、他にも多くの直接的なメリットがあります。

- モーションは、ユーザーを楽しい気分にさせます。アニメーションと他の視覚的なフィードバックは、驚きと直感の瞬間を生み出します。その楽しさによって、ユーザーはデバイスとアプリを好きになります。
- モーションは、操作のヒントを与えます。モーションには方向があり、前後に動いたり、コンテンツの内外に動いたりし、ユーザーが現在のビューまでどのように到達したかに関する最小限の "階層リンク" のような手掛かりを残します。たとえば、Panoramic Entrance アニメーションでは、アプリが開いたときにメイン ナビゲーションにユーザーの目を描き、ハブ コントロールを操作する方法に関する手掛かりをユーザーに与えることができます。
- モーションは、パフォーマンスが向上した印象を与えます。ネットワークが遅延したり、システムが動作を一時停止したとき、アニメーションによってユーザーが感じる待ち時間を短くできます。
- モーションは、個性を加えます。よく考えられた Windows Phone UI では、モーションを使って、アプリが今この場に関係があるという印象が生み出され、入れ子になった階層にユーザーが入り込んでいるという印象を和らげられています。
- モーションは、一貫性を高めます。切り替えによって、ユーザーは既によく知っているタスクとの類似性を引き出して新しいアプリケーションの操作方法を学習できます。
- モーションは、洗練さを高めます。アニメーションを使うと、ユーザーは電話がフリーズしているのではなく処理中であることを知ることができ、ユーザーが関心を持つかもしれない新しい情報を受動的に表示することができます。

モーション エコシステムの規則

- 各アニメーションはシステムの一部であるため、目立ちすぎるアニメーションがあってはなりません。
- 各アニメーションは、目的を果たしたらすぐに見えなくなります。
- アニメーションは、3D ではなく 2.5D で表示され、複雑なテクスチャ、照明、浮き出しは使用しません。たとえば、Tilt アニメーションは、タッチ ターゲットの背後にある奥行は表現せず、平面のみ表現します。これによりコンテンツが強調されます。
- UI 要素は、他の UI 要素の周りを動くことがあります。たとえば、オブジェクトがページ上で拡大された場合、アニメーションによってページが方向に拡大される必要があります。
- モーションは、ハードウェアとソフトウェアのどちらの操作によってもトリガーできます。切り替えは、あらゆる可能性に対応するように設計する必要があります。
- アクション可能な項目は、ユーザー操作に基づいて移動します。たとえば、Tilt アニメーションは、オブジェクトがタッチされていることを示すためにアクティブ化されます。
- 項目が、ユーザーの注意に値する重要度レベルに更新された場合、モーションを使ってユーザーに通知する必要があります。

モーションの方向

Windows Phone デバイスでは、"進む" ナビゲーションとハードウェアの戻るボタンが使われるため、次の図のマトリックスに示すように、切り替えとアニメーションは 2 方向または 4 方向に移動する可能性があります。この図は、Turnstile アニメーションを例として使い、ユーザーがアプリに入ることができる 4 つの方向を示しています。

ナビゲーション マトリックス

2 つまたは 4 つの特定の前後モーションを使うと、ユーザーはタスク フローで自分の位置を把握し続けることができます。

通常、アプリ設計のごく初期段階からモーションの役割を考慮に入れてください。設計プロセスの末期にモーションを追加すると、自然な方法で統合することが難しくなり、モーションには何度も使うことができるというメリットがある分、慎重なプロトタイプが必要になります。

イージング

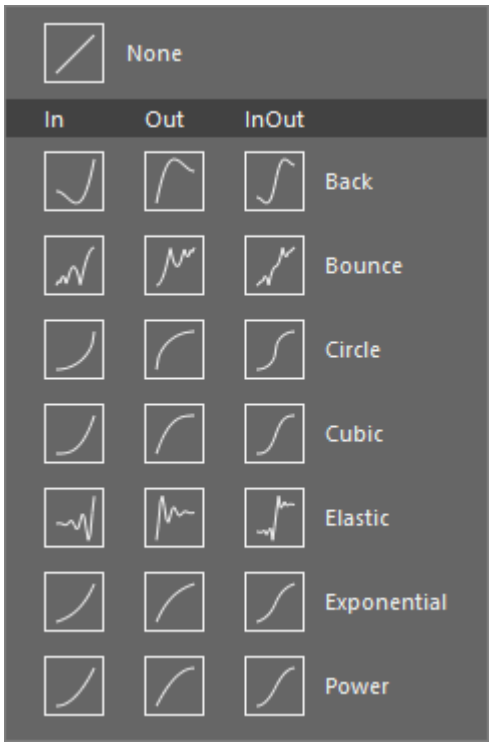
"イージング" とは、時間の経過と共にアニメーションの速度が変化する方法のことです。イージングを使うと、UI オブジェクトは現実の物体が動くような方法で動くことができます。落下するオブジェクトやロケット エンジンは停止状態から加速するため (その速度に "イージング イン")、画面から出るオブジェクトに適しています。回転するオブジェクトや閉まるドアは停止に向かって減速するため (その速度から "イージング アウト")、ビューに向かってアニメーション化されるオブジェクトに適した種類のモーションです。

曲線は、各種類のイージングの資格化に使われるプロファイルです。曲線は、X 軸で時間を示し、Y 軸でアニメーションの進行状況を示すグラフです。45 度の直線であるイージング曲線は、最初から最後まで速度が一定であることを意味します。角度が徐々に大きくなる曲線は加速を示しており (速度にイージング イン)、角度が徐々に小さくなる曲線は減速を示しています (速度からイージング アウト)。

直感に沿ったイージングを行うと、ユーザー エクスペリエンスが大きく変わる可能性があります。多くのラボ調査では、単にイージングが正しく調整されていなかったというためにマイナスの結果が出ました。最も一般的な種類のイージングを次に示します。

- イージング イン: オブジェクトが画面を離れる場合、イージング イン曲線を使うことで、張力を生み出してオブジェクトをすばやく飛ばします。指数曲線の形は、先頭に向かって平らになるため "通常" はこの種類のイージングを表していますが、イージング関数では主な要素はイージング インかイージング アウトかです。イージング関数では、"指数" などの用語は、曲線の程度を表しているにすぎず、イージング イン、イージング アウト、イージング イン/アウトのいずれの可能性もあります。

- イージング アウト: オブジェクトが画面に入る場合、イージング アウト曲線を使って滑らかにそっと停止します。対数曲線の形 (または、上部に向かって平らになる他の曲線) は、この種類のイージングを表しています。
- イージング イン/アウト: イージング イン/アウトは、オブジェクトのモーションが画面上で開始して終了するとき (特に動きが大きい場合) に適しています。S 字曲線は、この種類のイージングを表しています。



イージング関数

フィードバックの生成

ユーザーがタッチ ターゲットをタップしたときに、適切なフィードバックを生成するようにしてください。カスタム コントロールの表示状態を設計し、操作やアクティブ化の各段階でそれらを示します。ユーザーは、ボタンが押されたことや、コントロールが切り替えられたことを知ることができます。

注 Windows Phone でタッチ フィードバックに広く使われている方法はチルト効果です。これは、ボタンやリスト項目など、既定のスタイルのコントロールに組み込まれています。「PointerDownThemeAnimation」をご覧ください。

- **変更をすぐに示す:** 一部の設定を除き、コントロールは通常すぐに変更を引き起こすことでフィードバックを示す必要があります。

- **タッチ コントロールの操作に対する視覚的または聴覚的なフィードバックをすぐに返す:** すべてのアクションは、すぐに明確な結果を出す必要があります。応答は、後ではなくジェスチャが起きているときに生じる必要があります。良くない設計の例は、ユーザーが写真をフリックしたときに、ジェスチャが終わってから移動が生じることです。
- **時間のかかる処理にフィードバックを返す:** 処理が続いている間、何かが行われていることをユーザーに知らせる必要があります。コンテンツを使って進行状況を示したり、最終手段としてプログレス コントロールや直接通知を使うことができます。
- **ジェスチャに対する応答は、電話とアプリの間で一貫している必要がある:** Windows Phone SDK でタッチ コントロールを使うと、このトピックで説明しているタッチ ジェスチャとフィードバックが組み込みでサポートされるため、一貫性を保つことができます。カスタム タッチ コントロールを作る場合、同じような方法でジェスチャに応答する必要があります。

出力

Windows Phone アプリには、4 つの出力方法があります。

- ディスプレイ
- オーディオ出力ジャック
- 内部スピーカー
- バイブレーション

アプリを設計するときは、4 つの出力をすべて考慮に入れる必要があります。

サウンドとバイブレーション

Windows Phone では、サウンドとバイブレーションがユーザー エクスペリエンスに意味のある貢献をします。オーディオを、ユーザーにアクションを促す方法としてではなく、アプリのサウンドトラックとして考えましょう。アプリのオーディオを設計するときは、ミニマリズムの基本原則を適用し、邪魔なオーディオや不必要なオーディオを避けてください。優れたアプリでは、視覚的な設計を補ってサポートするために、オーディオが使われています。

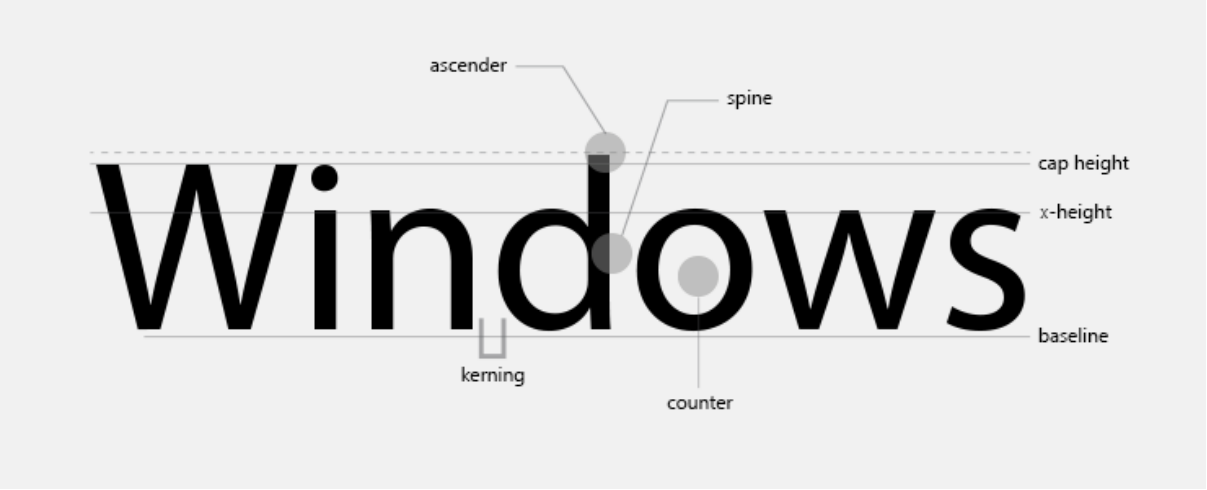
オーディオのガイドラインは次のとおりです。

- 理由のオーディオを避ける。
- 汎用的なサウンド効果を避ける。
- ユーザーの不安を駆りたてる、耳障りで驚くようなサウンドは避ける。

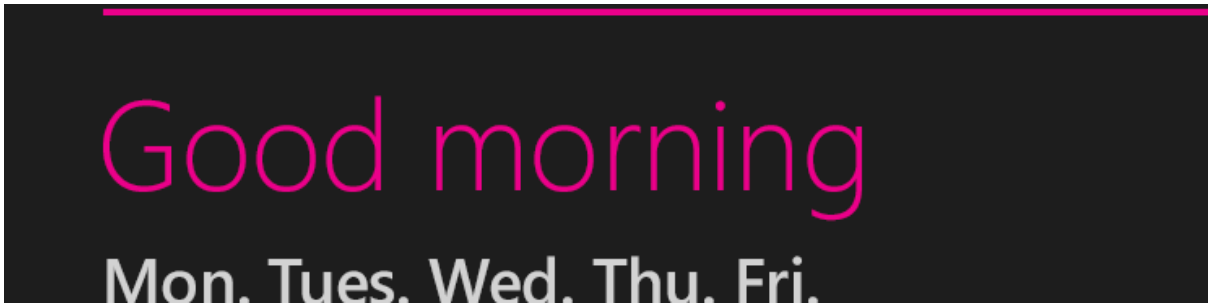
注 バイブレーション ユニットは、ユーザーが [設定] の [着信音+サウンド] でオン/オフにできます。アプリでこの設定を無効にすることはできません。

サウンド要素は、ユーザー エクスペリエンスに意味のある貢献をする必要があります。すべてのオーディオ アセットは、雑音が少なく、適切に編集され、ボリュームのバランスが取れたプロフェッショナル品質でなければなりません。

書体



メール、本、道路標識、メニューに書かれた価格、タイヤの空気圧のマーキング、道路脇のポールに掲示されたポスターなど、日常生活の中で文字を目にする機会はたくさんあります。私たちの周りには、エンターテインメント性のあるテキスト、情報を提供するテキスト、教育的なテキスト、意味のあるテキストが氾濫しています。その中心となるのが、タイポグラフィとレイアウトです。



イメージは魅力的で目を引くビジュアル効果を作るのに対し、タイポグラフィは、コンテンツを読者に伝え、優れた設計を保証するものです。美しいタイポグラフィを作るには、次に点に注意してください。

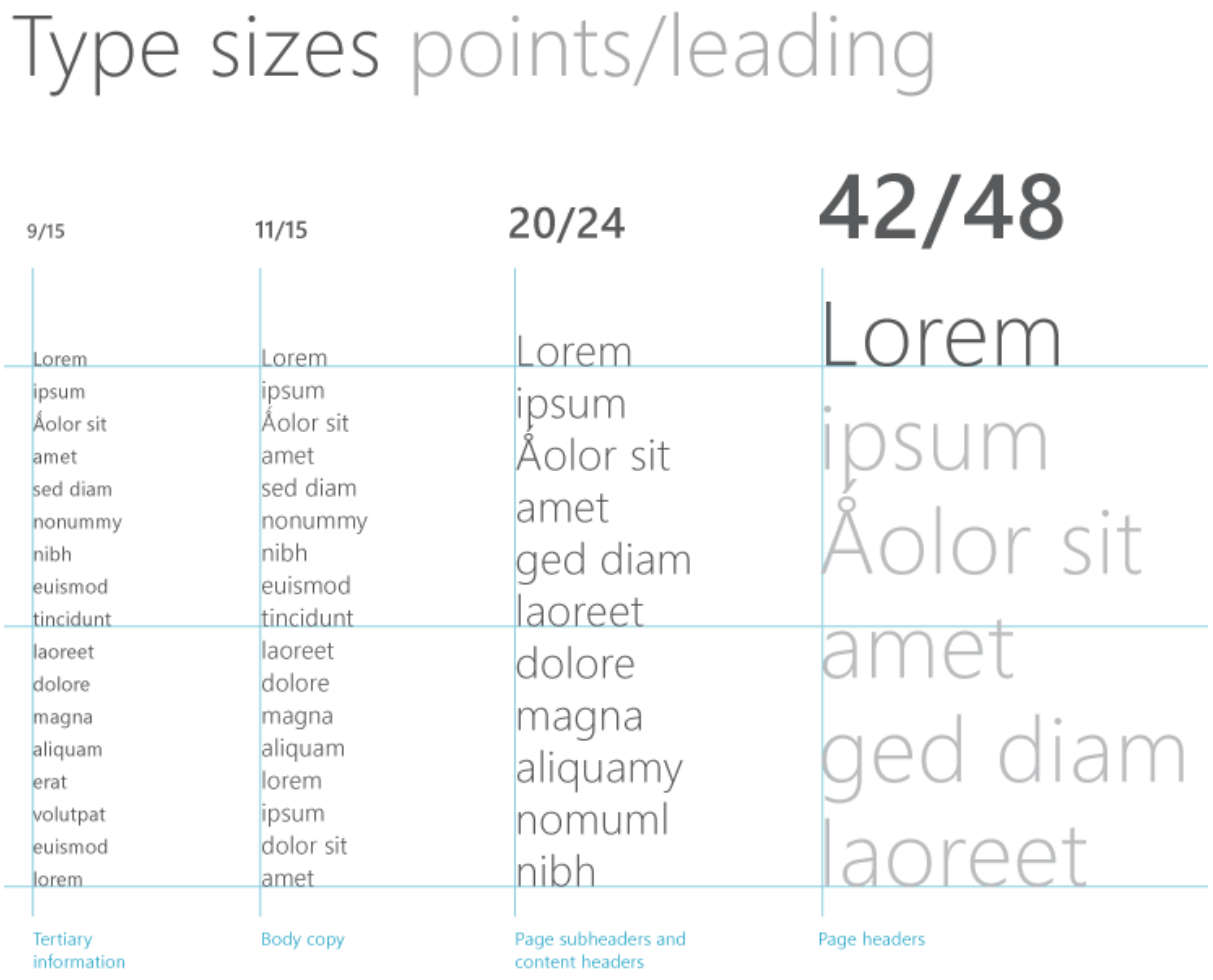
- フォントの選択
- 空白
- 階層
- 行の高さ
- テキスト サイズ

これらの側面を適切に適用すると、明快で意図的、容易にナビゲートできる、高品質で有効な設計になります。

Windows のタイポグラフィ

Windows では、"コンテンツ優先" の原則に基づき、不要な要素や UI が排除されました。この最小限のものしかない環境では、タイポグラフィがより際立ち、ナビゲーションにとって重要になります。

書体見本 (type ramp) に基づく詳しいフォントのガイドラインが確立されています。書体見本 (type ramp) は、ヘッドラインからの本文までの重要なデザインの関係を確立し、異なるレベル間の明快でわかりやすい階層を保証します。ユーザーは、情報を見つける場所やページの解析方法をすぐに理解できます。

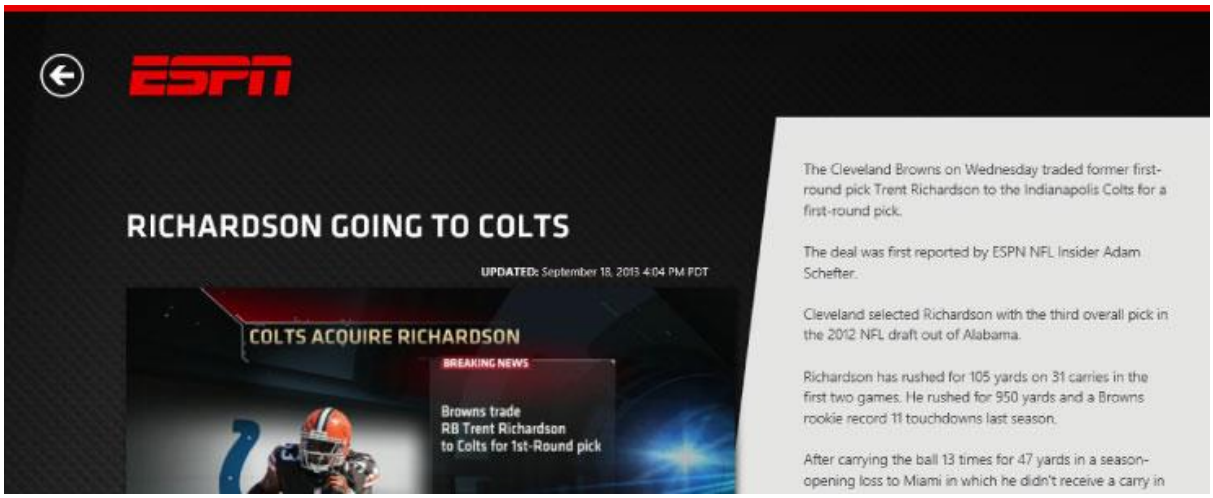


私たちは、ダッシュ、省略記号、大文字/小文字の使用、アポストロフィ、引用符を含めて、タイポグラフィの詳細についても慎重に検討しました。これらの項目の検討と書体見本 (type ramp) により、Windows エクスペリエンスの一貫したビジュアルとブランドが確立されます。これは、細部にこだわり持って優れたタイポグラフィを追求した結果です。

タイポグラフィの使い方

タイポグラフィをうまく使うと、コンテンツを伝える以上の効果を得ることができます。また、ブランドの独自性を強化したり強調したりすることもできます。それでは、いくつかの例を見ていきましょう。

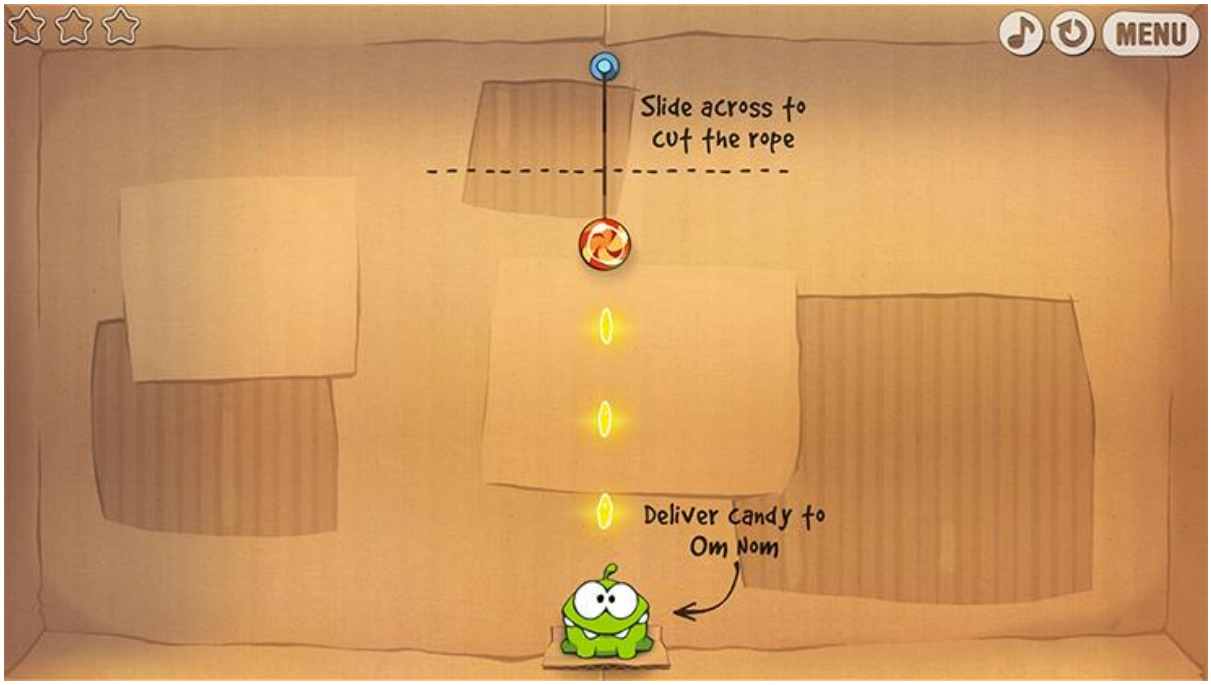
この ESPN アプリには、ヘッドラインに Font Bureau の Benton Sans が効果的に使われ、アプリの特定の外観が確立されています。一般的かつ標準のフォントに代えて Benton Sans を使うことで、アクティブでパワフル、専門的な印象がアプリにもたらされています。これにより、他のアプリとの差別化が図られ、ESPN のブランドがサポートされます。



Expedia は、より単刀直入にブランドを強調しています。このアプリでは、Expedia は、ロゴに密接に同調するフォントを使っています。これにより、ヘッダー テキストの行を見たユーザーに使っているアプリを思い起こさせる効果に加え、アプリ全体で一貫したブランドが保証されます。

Disney の Princess Dress-up Sticker Book アプリでは、各王女のストーリーを伝える物語のセクションに対して、League of Moveable Type の Goudy Bookletter 1911 が使われています。この書体は空想的で親しみやすいため、対象読者に親しみのある印象をテキストに与えています。さらに、このアプリでは、行間を十分に取った大きな文字を使って、低年齢の読者にとっても物語の要素が読みやすくなるように工夫しています。

Cut the Rope は、キャンデーが大好物なキュートで小さなモンスターで知られる、よく知られた楽しいアプリケーションです。このように陽気で人の心を引き付けるテーマを持つアプリでは、モンスターにキャンデーを与えるための手順を解説する重要な説明に、楽しくかつ読みやすい書体を使う必要があります。アプリの作成者が選んだフォントは、Fontheadの Good Dog Plain です。のんびりとした手書き風のこのフォントは、製品のブランドを効果的に表現します。ここには退屈なフォントはありません。



アイコン

アイコンは、説明なしに伝えるべきことを伝えるものであり、ナビゲーションの要素としても、非常に重要です。アイコンとは、もともと象徴的な要素であり、テキストの代わりに意味を伝えます。つまり、可能な限り速く、効率的にユーザーを次のステップに導く役割があります。アイコンは生産性の構成要素であり、UI におけるナビゲーションのビジュアル基盤となります。

Windows のアイコン

Windows のアイコンは、グラフィックスと記号です。わかりやすい明確な線と単純な図形が使われます。タッチ操作主体のシステムのナビゲーションに不可欠で、ユーザーが UI 内を移動するのを助けます。決して、箇条書きの代わりに使わないでください。

Windows のアイコンは、最小限の労力と詳細を使って概念を抽象化、簡略化するものです。本来グラフィックスで表され、フラットかつ単色であるため、クロームよりコンテンツという考えを強調したものです。

視点

Windows のアイコンは 2 次元の記号による図形です。アイコンはボリュームまたは深さをさりげなく示唆できますが、一般的に、遠近を表す線や角度の追加は避けます。ほとんどのアイコンは 1 つの概念、あるいはオブジェクトを表します。アイコンが複数のオブジェクトを表す場合、後ろのオブジェクトが前のオブジェクトに "ノックアウト" されることになります。

オブジェクトの "ノックアウト " とは、 アイコンの要素が別の要素に重なり、その重なりによって背景の色に一致するギャップが生まれることです。

色

Windows のアイコンのメインの色は純粋な白です。白は、色鮮やかな背景を持つアプリ タイルやアプリ バーなど、ほとんどの色のコンテキストにマッチする単色です。

詳細

Windows では、線の太さ、角度、角、"ノックアウト" などの細部に細心の注意が払われています。そのため、分類された多数のグリフ アイコンがファミリとしてまとめられて保持されています。 タッチ システムの線の太さは、弱々しさを感じさせてはなりません。小さなアイコンの場合、1 ピクセル違うだけで、線またはオブジェクトが "大きすぎる" または " 小さすぎる" という印象を与えることがあります。タッチ アイコンの場合は、常に大きい方を選ぶようにしてください。詳細 (ノックアウトを含む) は、少なくとも 2 ピクセルにする必要があります。

高 DPI に合わせたスケーリング

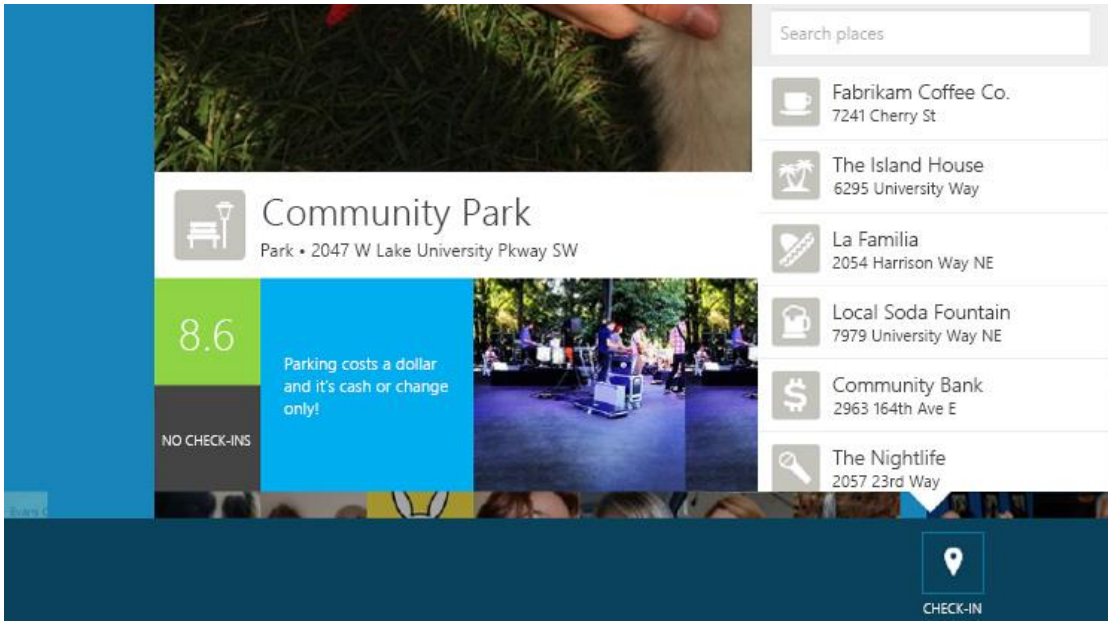
Windows のアイコンは、3 つの主要なスケール プラトーを対象としています。基本のアイコン サイズは 1x です。高 DPI サイズは、1.4x と 1.8x です。高いスケール プラトーにスケーリングすると、ピクセルが部分的になり、端数が切り捨てられて最も近いピクセルに丸められます。

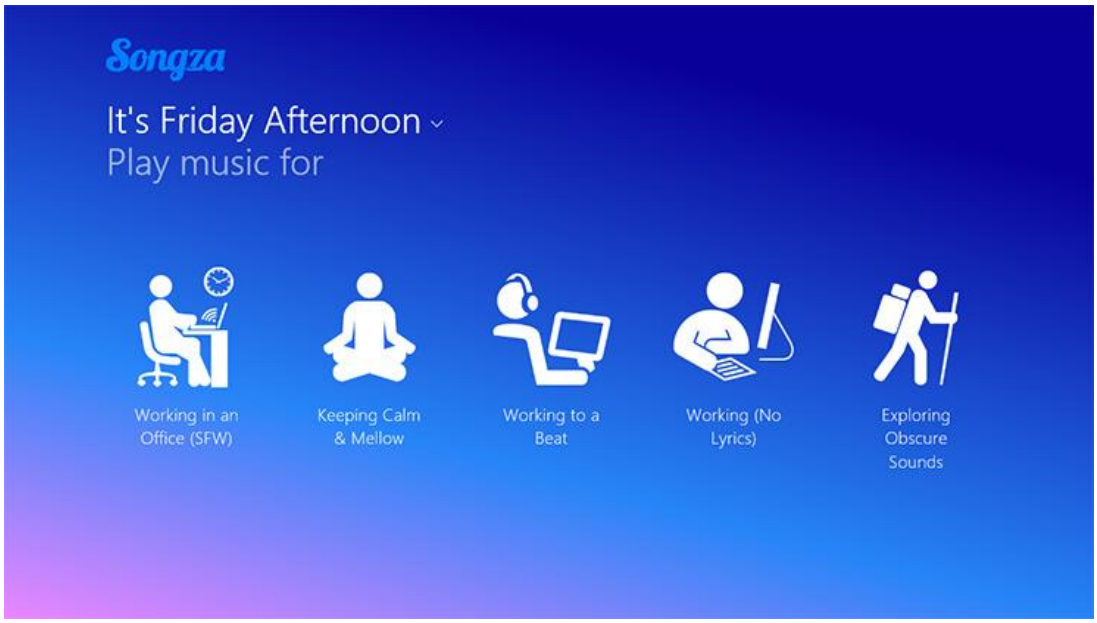
制作

Windows アイコンは、Adobe Illustrator または Expressions Design を利用して、すべてベクターで描画することをお勧めします。ピクセル グリッドへのベクター図形のスナップは、グリフ システムでは不可欠です。Windows のアイコンのエッジは、鮮明でくっくりしたものにする必要があります。

アイコンの使い方

アイコンはシンプルであるため、個性を追加しやすい要素です。アイコンのシステムを見ると、ブランドがそれ自体を識別するだけでなく、ユーザーとの一体感を作り出すしくみがわかります。アイコンは、最も基本的なレベルで描画されるビジュアルの印象により、ストーリーを提供します。アイコンにより、ごくシンプルな方法でアプリを差別化できます。統一感のある場所アイコンを持つ Foursquare アプリや、シナリオがアイコンとして描画されている Songza アプリを参考にしてください。





アイコンは、機能を提供するだけにとどまりません。ユーザーをアプリの全体的なビジョンに結び付け、アプリに対する好ましい感情を呼び起こすことすらできます。Songza は、アイコンにアプリケーションの適切な属性を割り当てています。インターフェイス上の効率性を維持しつつ、ブランドの個性が光っています。

色

色を使って、意図を伝えることや雰囲気や感情を表すこともできます。色でアプリ内のコンテンツを強化することができます。また、全体的なレイアウトにビジュアル構造を加えることができます。さらに、ブランドの価値と特徴を伝えることができます。

Windows での色の使い方

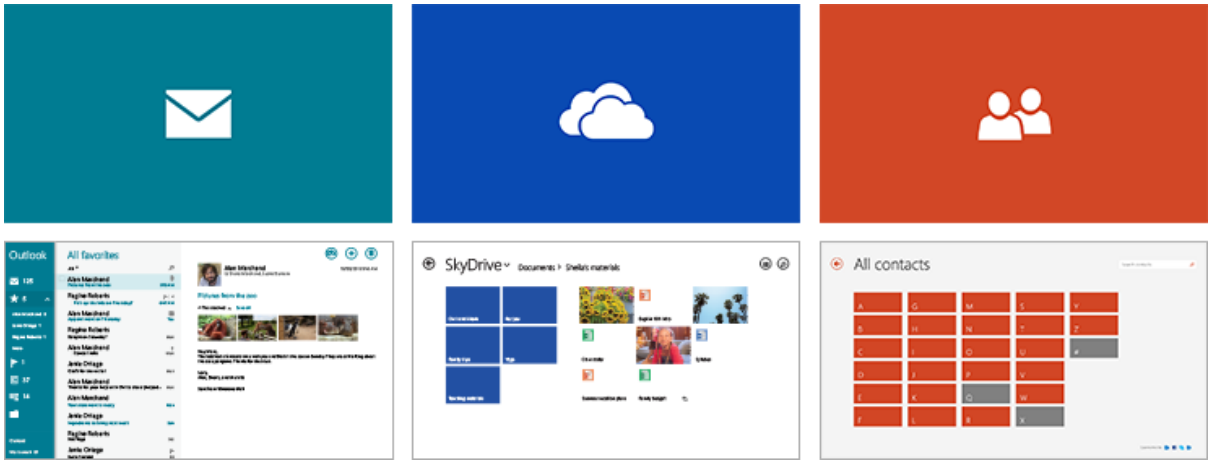
Windows における色使いは、アプリをエネルギッシュで生き生きと感じられるようにすることがねらいです。色は濃く、力強く、真にデジタル化され、表示テクノロジで利用できる色の全域に対応しています。Windows では、均一な単色を使うことがほとんどですが、透過、レイヤー、ブレンドなどの深みを加えるさまざまな手法も採用しています。

Windows 8.1 では、システム カラーとアプリの色は分かれています。システム カラーは、ユーザーが選ぶ色であり、Windows でシステム内のプライマリ UI サーフェス (スタート画

面の背景など) に適用される色です。アプリの色は、開発者が選ぶ一意の色であり、システム カラーとは関係ありません。アプリの色は、アプリのブランドと独自性を表します。

次の図に、スタート画面に適用されたさまざまなシステム カラーを示します。

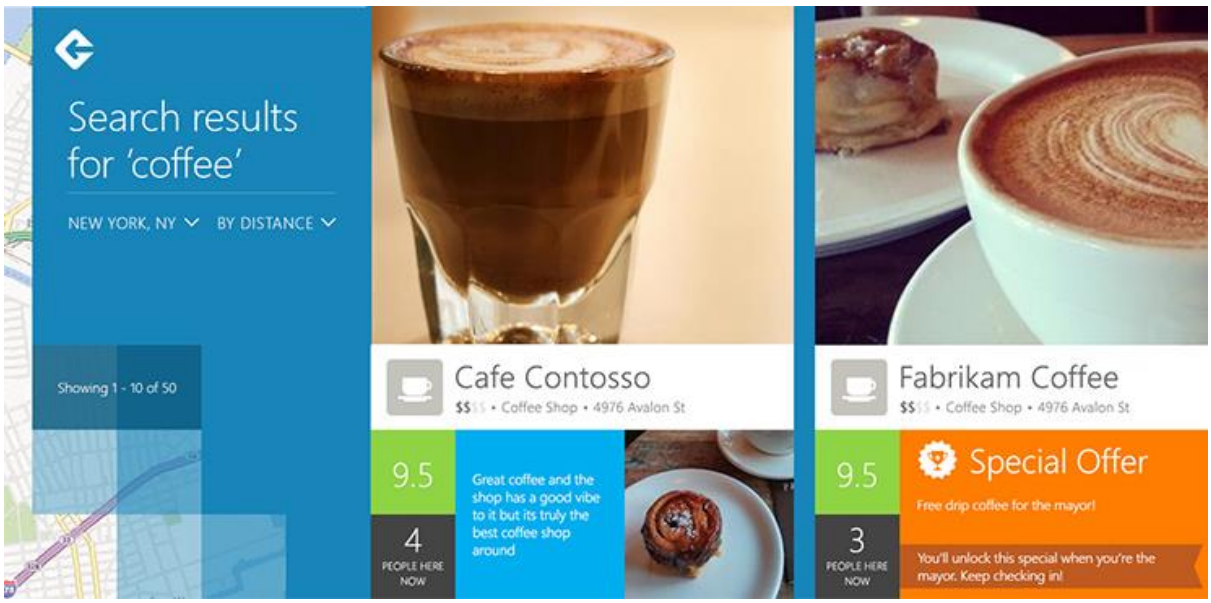
次の図に、3 種類のアプリのテーマ カラーを示します。アプリの色は、ブランドをアピールするためにアプリ全体に適用されます。



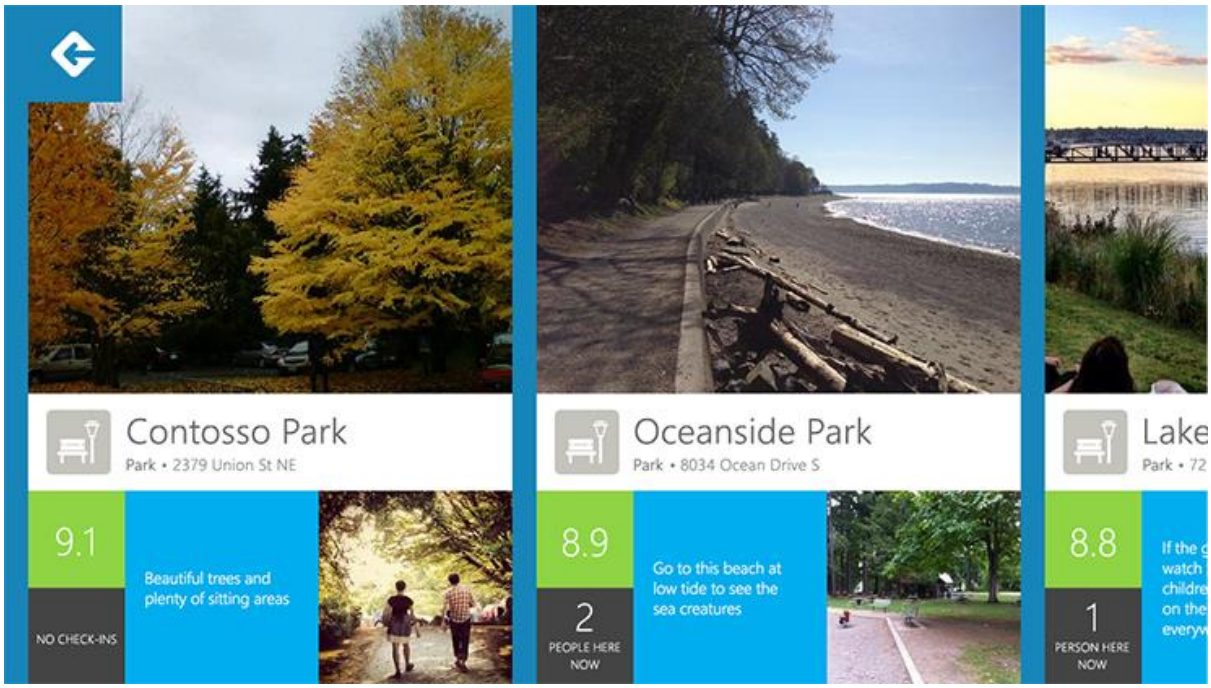
色の働き

Windows 8.1 では、アプリの特徴を反映する色を選ぶことで、アプリのブランドに独自性を与えることができます。ブランドをさらに強化するために、サポートされる追加の色を使うことができます。いくつかの例を見てみましょう。

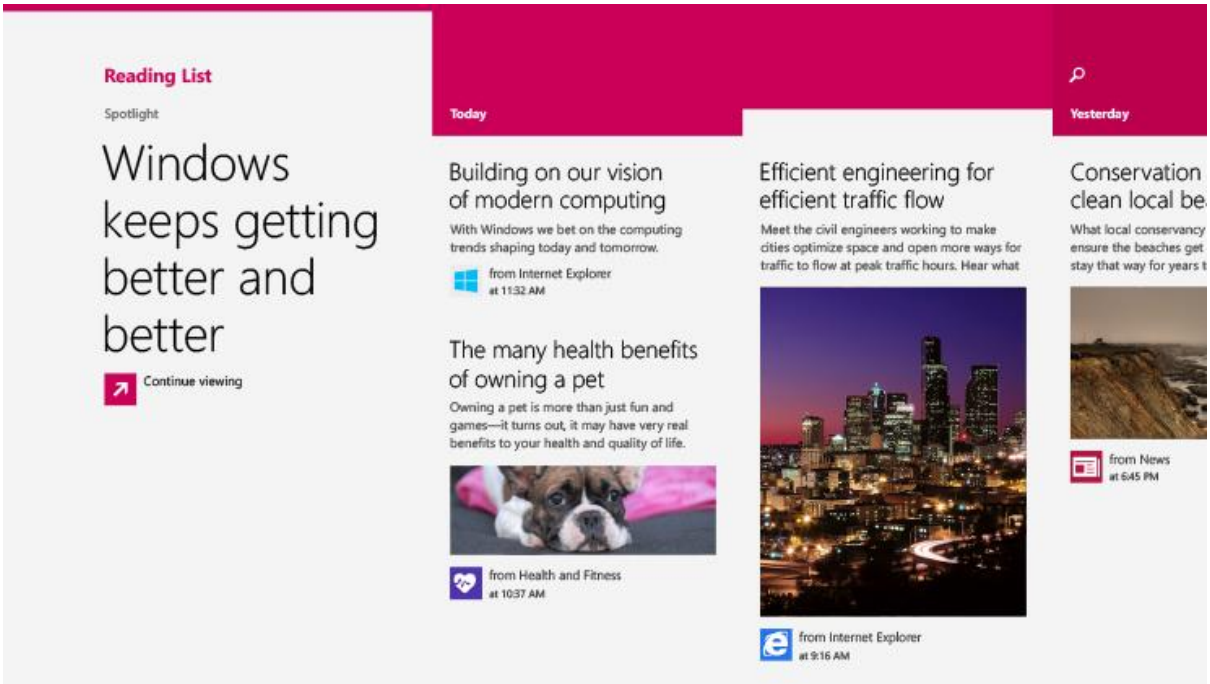
この Foursquare アプリでは、青色と緑色の一貫した色の組み合わせを使って、Foursquare ブランドを印象づけ、ナビゲーションがスムーズに行われるようにしています。



見た目のシンプルさとタイル内の大胆な色使いが、ユーザーに探索したいと思わせます。色により、アプリのさまざまな情報レベル (ヒント、評価、人気など) で直感的に移動先を見つけることができます。



リーディング リスト アプリは、色でアプリに独自性を与えているもう 1 つの例です。この場合、アプリのテーマ カラーである赤はコンテンツの新しさを伝えています。アプリの上部にあるカラー バーは、ユーザーがコンテンツを保存した日付が古いほど徐々に色が濃くなっています。これにより、色を使ってアプリの独自性を高めると共に、ユーザーに追加の情報を提供しています。



Windows Phone 用テーマの設計の決定

テーマは、ユーザーが選択した背景とアクセントの色の組み合わせで、Windows Phone の視覚要素をカスタマイズするものです。テーマには色のみが含まれ、フォントやコントロールのサイズのような他の要素は変更されません。

濃色または淡色の 2 つの背景色があり、複数のアクセント カラーから選択できます。

Windows Phone アプリは、選択されたテーマを自動的に使用するため、システム コントロールと UI 要素がプラットフォーム全体で一貫して表示され、シームレスで安心できるユーザー エクスペリエンスが実現します。

独自のテーマを設定する場合は、アプリ内のテーマを上書きできます。カスタム テーマは、ブランド化されたアプリに品質、独創性、または安心感を与えることができます。アプリのビジュアル デザインのカスタマイズ量を事前に決定し、それにしたがって要素を計画します。自分のリソースを提供し、テーマのプロパティを上書きできますが、テーマを無効にすることはできません。

テーマの機能

テーマは、Windows Phone の視覚要素をカスタマイズするために使用される一連のリソースです。テーマ要素は [テーマ] の下の [設定] メニューでユーザーが設定できます。

Windows Phone SDK のテーマ リソースを使うと、ネイティブ UI の外観を維持するアプリを作成できます。テーマのプロパティはコードで直接アクセスでき、テーマのプロパティの値を、ブランドまたは配色に合うように明示的に変更できます。

背景

背景はアプリの機能要素の背後に表示されます。これは、ブランドまたは他のカスタム アートを表示してアプリのトーンを設定するものです。背景は動的なものにもできます。新しい、または変化する情報や画像を特定の間隔で表示できます。たとえば、ハブには、すべてのパネルに及ぶ背景画像を含めることができます。通知を使って、アプリのタイルの背景画像をいつでも更新できます。

サポートされているアクセント カラー

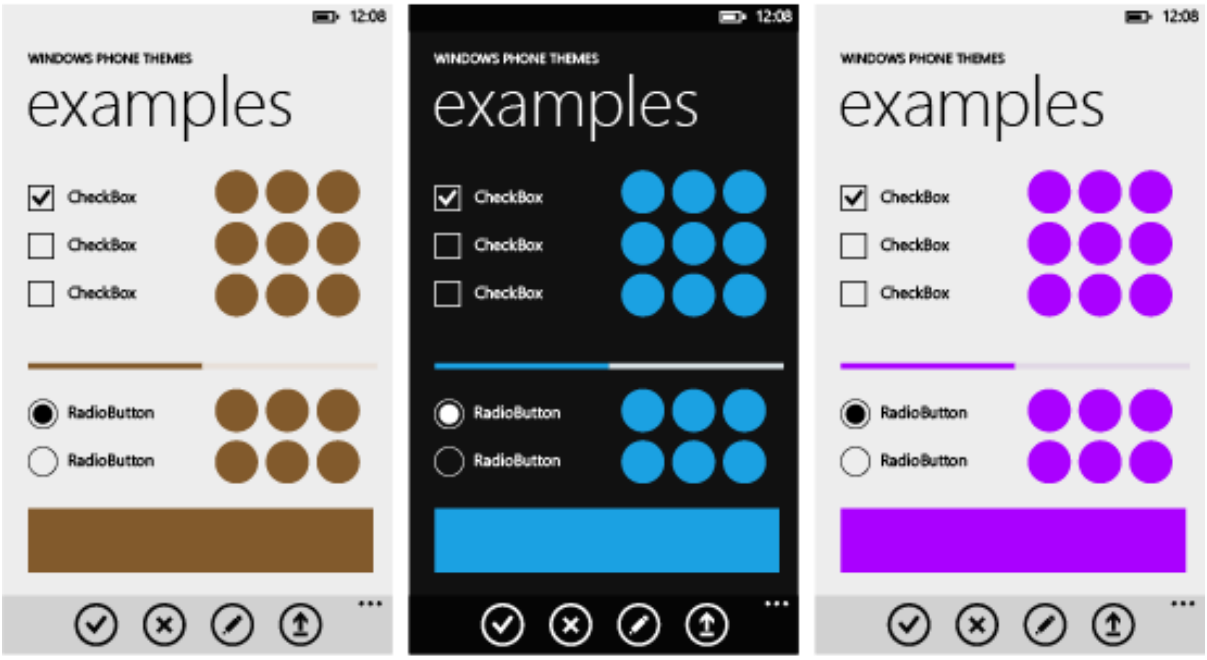
Windows Phone では、ユーザーはさまざまなアクセント カラーから選ぶことができます。既定のテーマは青いアクセント カラーの濃色の背景ですが、通信事業者や電話製造業者がこの設定を上書きする場合もあります。

ACCENT COLOR	RED, GREEN, BLUE	HEX	WINDOWS PHONE 7.5	WINDOWS PHONE 8
Lime	164, 196, 0 162, 193, 57	#FFA4C400 #FFA2C139	✓	✓
Green	96, 169, 23 51, 153, 51	#FF60A917 #FF339933	✓	✓
Emerald	0, 138, 0	#FF008A00		✓
Teal Viridian	0, 171, 169	#FF00ABA9	✓	✓
Cyan Blue	27, 161, 226	#FF1BA1E2	✓	✓
Cobalt	0, 80, 239	#FF0050EF		✓
Indigo	106, 0, 255	#FF6A00FF		✓
Violet Purple	170, 0, 255	#FFAA00FF	✓	✓
Pink	244, 141, 208 230, 113, 184	#FFF472D0 #FFE671B8	✓	✓
Magenta	216, 0, 115	#FFD80073	✓	✓
Crimson	162, 0, 37	#FFA20025		✓
Red	229, 20, 0	#FFE51400	✓	✓
Orange Mango	250, 104, 0 240, 150, 9	#FFFA6800 #FFF09609	✓	✓
Amber	240, 163, 10	#FFF0A30A		✓
Yellow	227, 200, 0	#FFE3C800		✓
Brown	130, 90, 44 160, 80, 0	#FF825A2C #FFA05000	✓	✓
Olive	109, 135, 100	#FF6D8764		✓
Steel	100, 118, 135	#FF647687		✓
Mauve	118, 96, 138	#FF76608A		✓
Sienna	160, 82, 45	#FFA0522D		✓

BLUE TEXT denotes Windows Phone 7.5 names and values

ヒント: 一部のデバイスには、通信事業者が指定した追加のアクセント カラーがプレインストールされている場合があります。

アプリではこの追加のカラーを把握し、上の表にあるシステム全体の色について憶測を立てないようにする必要があります。たとえば、標準のアクセント カラーをチェックしてから特定の操作を実行するコードを作成しないようにします。追加のカラーによって操作が失敗する可能性があるためです。



配色

色に関するよくある間違い

ユーザーがシステムを淡色テーマ モードにしても、ハイライト カラーが表示されるようにしてください。淡色テーマと濃色テーマの両方でアクセント カラーをテストします。ユーザーは複数のテーマから選択できるため、UI に色付きの要素を追加する場合は、可能な色の組み合わせを考慮してください。色の組み合わせについては、グラフィック デザイナーに相談することもできます。

重要な注意: Windows Phone には、淡色と濃色の 2 つのシステム テーマが含まれています。アプリではセカンダリ アクセント テーマを使うことをお勧めします。ただし、ユーザー設定のシステム テーマは Windows Phone SDK で使用できるすべてのコントロールに自動的に適用されるため、視覚デザインの一部が背景に隠れる場合があります。

ユーザーが Windows Phone のシステム テーマを変えると、テーマの色のみが変更されます。フォントやコントロールのサイズなど、他の要素は動的に変更されません。

ただし、他の Windows Phone テーマ リソースを使用して、フォントやフォント サイズなどのプロパティを変更できます。これらのリソースについて詳しくは、「Windows Phone のテーマ リソース」をご覧ください。

テーマ要素の作成

カスタム テーマ要素を作成する場合は、Windows Phone のメインの UI パラダイムであるナビゲーションのフレームとページ モデルに合うように設計します。フレームとページを使用すると、ユーザーはリンクを使ってコンテンツの複数のページ (または画面) を前に進み、ハードウェアの戻るボタンを使って後ろに戻ることができます。フレーム、ページ、画面について詳しくは、「Windows Phone 用のナビゲーション、向き、ジェスチャ」をご覧ください。

基本の配色を選びます。

ブランドにさりげなく影響するように色を結合する方法を考慮してください。Windows Phone では、明確さと見やすさを確保するために、控えめでハイ コントラストな色の組み合わせを使用します。派手な色は慎重に使います。色を決定できない場合は、中間色または単色のトーンにします。

実装

ユーザーが選択したシステム全体のテーマは変更できません。アプリ内のテーマのみ変更できます。

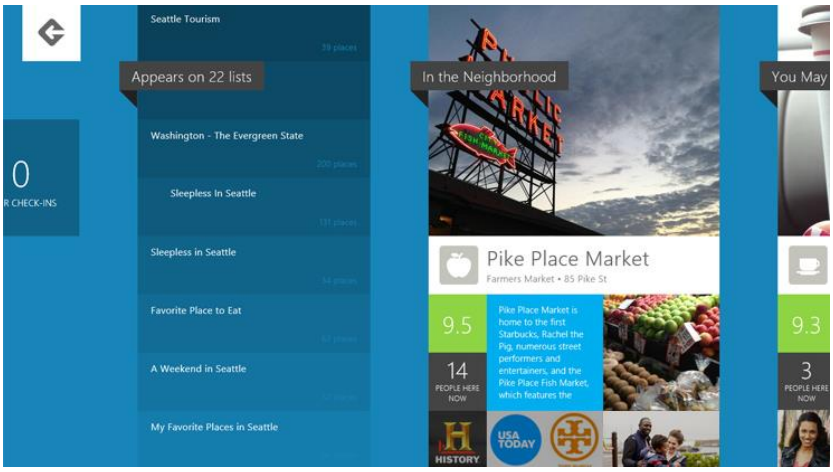
一般に、アプリでは、ユーザーが選択したテーマとアクセント カラーの選択を尊重する必要があります。ただし、これらのテーマと色を切り替えることがアプリの特定のブランドまたはカラー パレットを阻害する場合は、1 つのテーマのみ (淡色または濃色) にアプリをロックできます。これにより、システム全体の色の設定から、この色は切り離されます。

アプリ内の色と背景を総合的にコントロールする場合は、アプリをシステム テーマから切り離し、ユーザーの選択したテーマやアクセント カラーとは関係なく、ユーザー エクスペリエンスを総合的にコントロールしてアプリ内で色を行き渡らせることができます。

Windows Phone のさまざまなテーマ リソースと、それらをアプリで構成する方法については、「Windows Phone のテーマ リソースを適用する方法」をご覧ください。

インタラクション と UX

ナビゲーション パターン



Windows ナビゲーション パターンとしてフレームワークを提供しますが、技術革新を奨励しています。実績のあるパターンに基づいて創造性を発揮してください。

コマンド パターン

キャンバス、アプリ バー、チャーム、メニューを使って、ユーザーがアプリを制御できるようにします。

入力 パターン

ユーザーがアプリを操作できるさまざまな方法をご確認ください。優れたタッチ操作のデザインについて詳細をご覧ください。

ナビゲーション パターン

ユーザーが簡単かつ直感的に移動できるように、Windows アプリまたは Windows Phone アプリのコンテンツを整理します。

ナビゲーション パターン (Windows ストア アプリ)

ユーザーが簡単かつ直感的に移動できるように、Windows ストア アプリのコンテンツを整理します。適切なナビゲーション パターンを使うと、画面に常に表示されるコントロールを制限できます。これにより、ユーザーは、現在のコンテンツに集中できます。

Windows ストア アプリのナビゲーションは、階層システムとフラット システムという 2 つのパターンに基づいています。アプリは、いずれかのパターンを使うか、両方のシステムを組み合わせて使うことができます。

階層パターン

このパターンは、Windows ストア アプリを軽快かつ柔軟にします。コンテンツのコレクションが大きいアプリや、ユーザーがコンテンツのさまざまなセクションを探索するアプリに最適です。 多くの Windows ストア アプリでは、ナビゲーションの階層システムが使われます。

ハブ ページは、ユーザーの アプリへの入り口です。コンテンツが横方向に スクロールできるビューに表示され、 新しい情報や入手可能な項目が ひとめでわかります。

ハブはさまざまなカテゴリの コンテンツで構成され、それぞれがアプリの セクション ページに対応付けられています。

セクション ページは、アプリの 第 2 レベルです。シナリオと セクションに含まれる コンテンツを最適な形式で 表示できます。

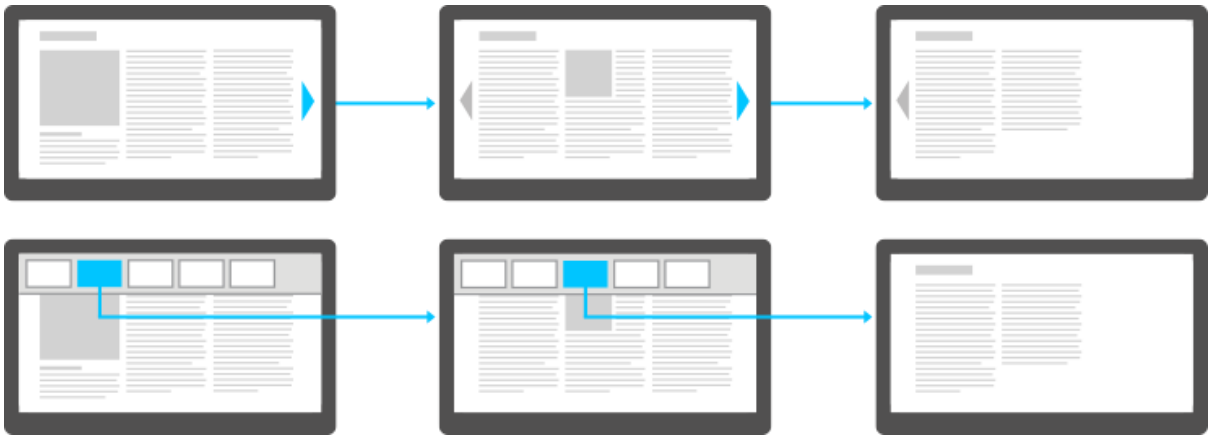
セクション ページは個別の項目で構成され、それぞれの項目に詳細ページがあります。 セクション ページでは、グループ化とパノラマ スタイルのレイアウトも使うことができます。

詳細ページは、アプリの 第 3 レベルです。個々の項目の詳細が 表示され、表示形式は コンテンツの タイプによって 大きく異なります。

詳細ページは、項目の詳細または機能で構成されます。 詳細ページには、多様な情報や、画像やビデオなどの単一のオブジェクトが含まれます。

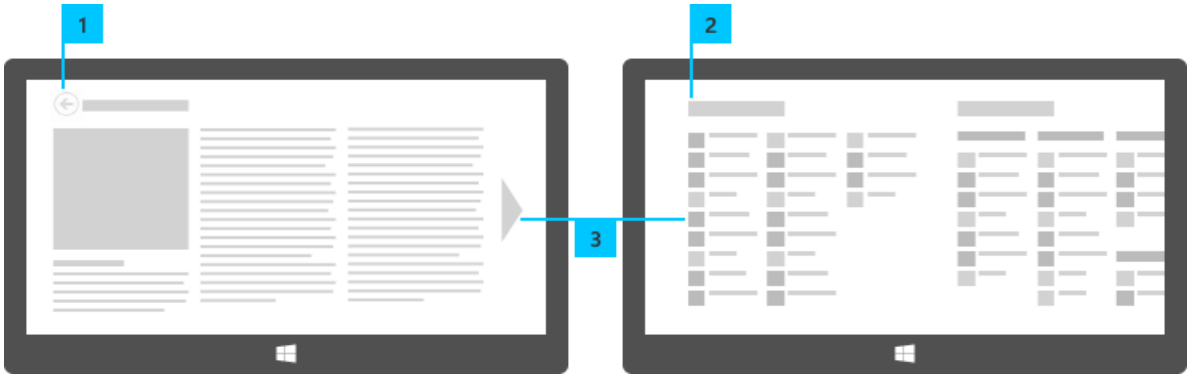
フラット パターン

多くの Windows ストア アプリは、フラットなナビゲーション パターンを使います。ゲーム、ブラウザー、ドキュメント作成アプリなどのアプリで、このパターンを使って、すべて同じ階層レベルにあるページ、タブ、またはモード間をユーザーが移動できます。階層パターンとは異なり、固定された [戻る] ボタンやナビゲーション スタックがないため、ページ間の移動は一般にコンテンツ内のダイレクト リンクを使うか (下の最初の例)、ナビゲーション バーを使います (下の 2 つ目の例)。



このパターンは、少数のページまたはタブ間での 高速切り替えがある中心的なシナリオに最適です。たとえば、Internet Explorer、電子ブック、ゲームのような Web ブラウザー アプリです。

キャンバス上のナビゲーション



1. ヘッダーと戻るボタン

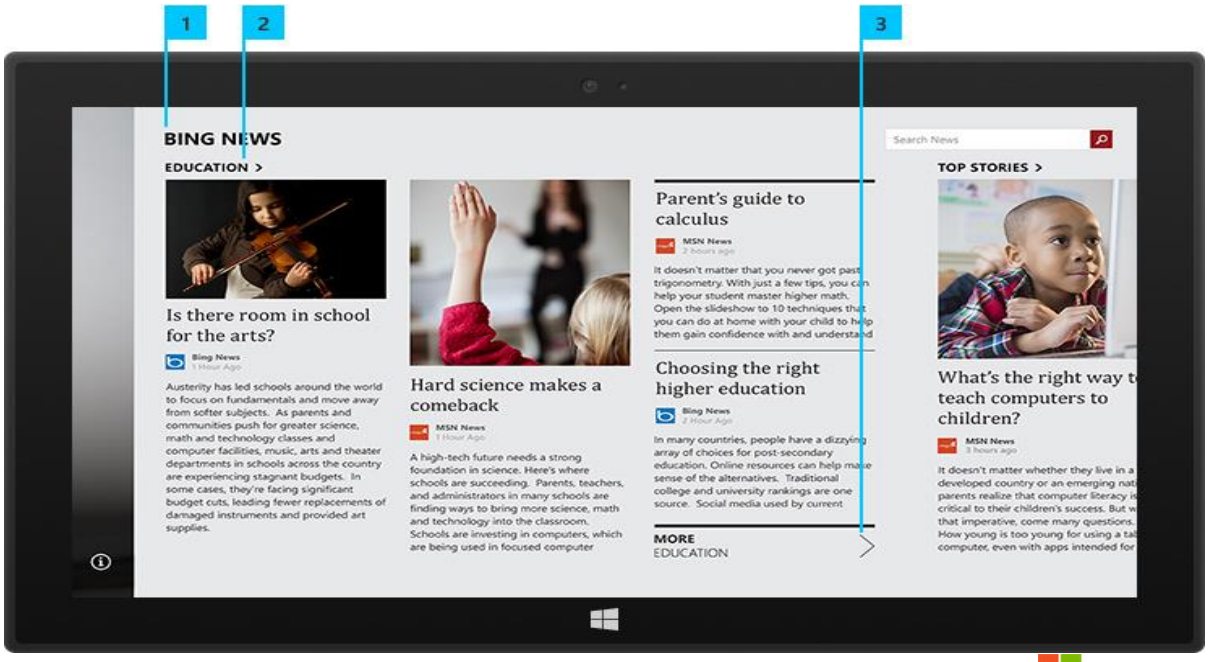
   ヘッダーは現在のページのラベルで、移動先を見つけるときに便利です。 戻るボタンを使って、元の場所にすばやく戻ることができます。

2. セクションまたはカテゴリ ラベル

   これらのラベルを使うと、コンテンツのさまざまなセクションまたはカテゴリに移動できます。

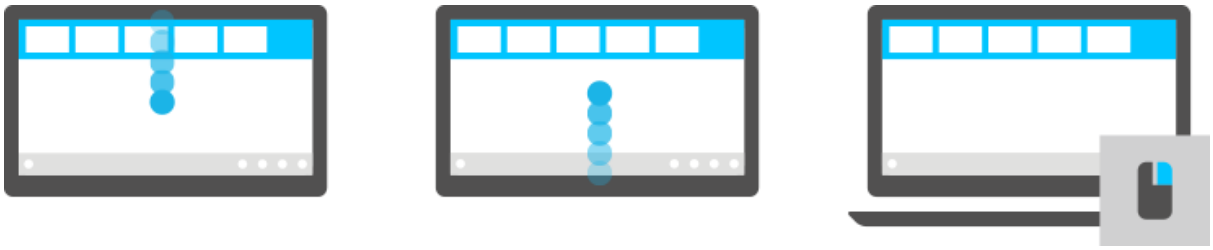
3. その他のターゲット

   ナビゲーション要素として、タイル、矢印、ボタン、検索結果、その他のカスタマイズされたターゲットを使うことができます。いくつかのゲームでは、興味深い図形によるナビゲーション要素のサンプルを見ることができます。



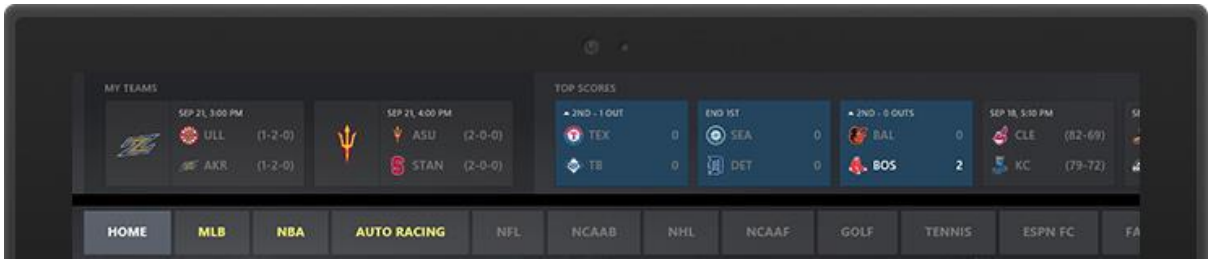
上側のアプリバー

画面の上端または下端をスワイプすると、アプリ バーが表示されます。上側のアプリ バーは、ナビゲーション バーとも呼ばれます。上側のアプリ バーにナビゲーション要素を配置して、より多くの画面領域をアプリのコンテンツが使えるようにすることができます。また、アプリを使っているときに頻繁にナビゲーション要素が必要になり、ナビゲーション要素がキャンバスにあってもアプリのエクスペリエンスに悪影響がない場合は、ナビゲーション要素をキャンバスに配置できます。ナビゲーション要素が上側バーまたはキャンバスのいずれに属するかを決定します。

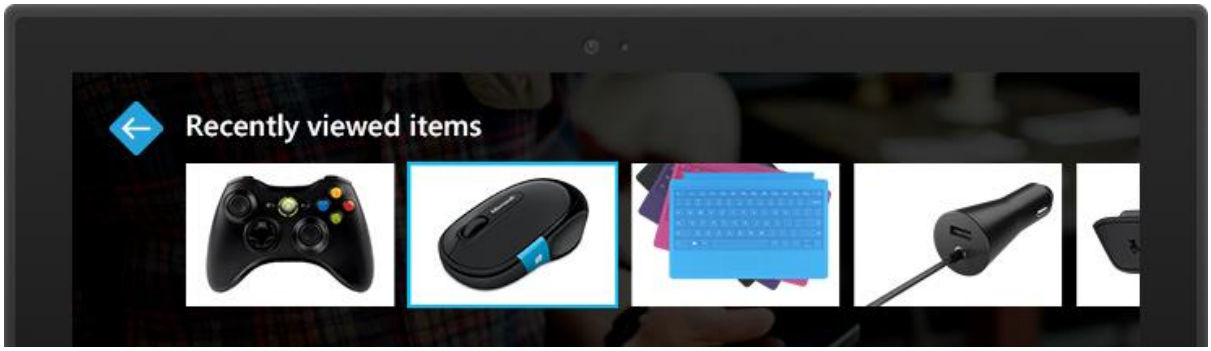


一般に、セクションまたはカテゴリ ラベルは、Hulu Plus のように、ナビゲーション バーにあります。

多くのアプリは、ショートカットを提示するために上側のアプリ バーを使っていました。たとえば、ESPN アプリでは、ユーザーは上側のアプリ バーのセクション ラベルの上にあるスコアボードをクリックして、ゲームキャストのページに移動できます。



上側のアプリ バーは、ユーザーにターゲット ページのコンテンツのプレビューを提示することもできます。次のショッピング アプリのサンプルでは、製品の詳細ページに移動する前に、アプリ バーで製品イメージを確認できます。

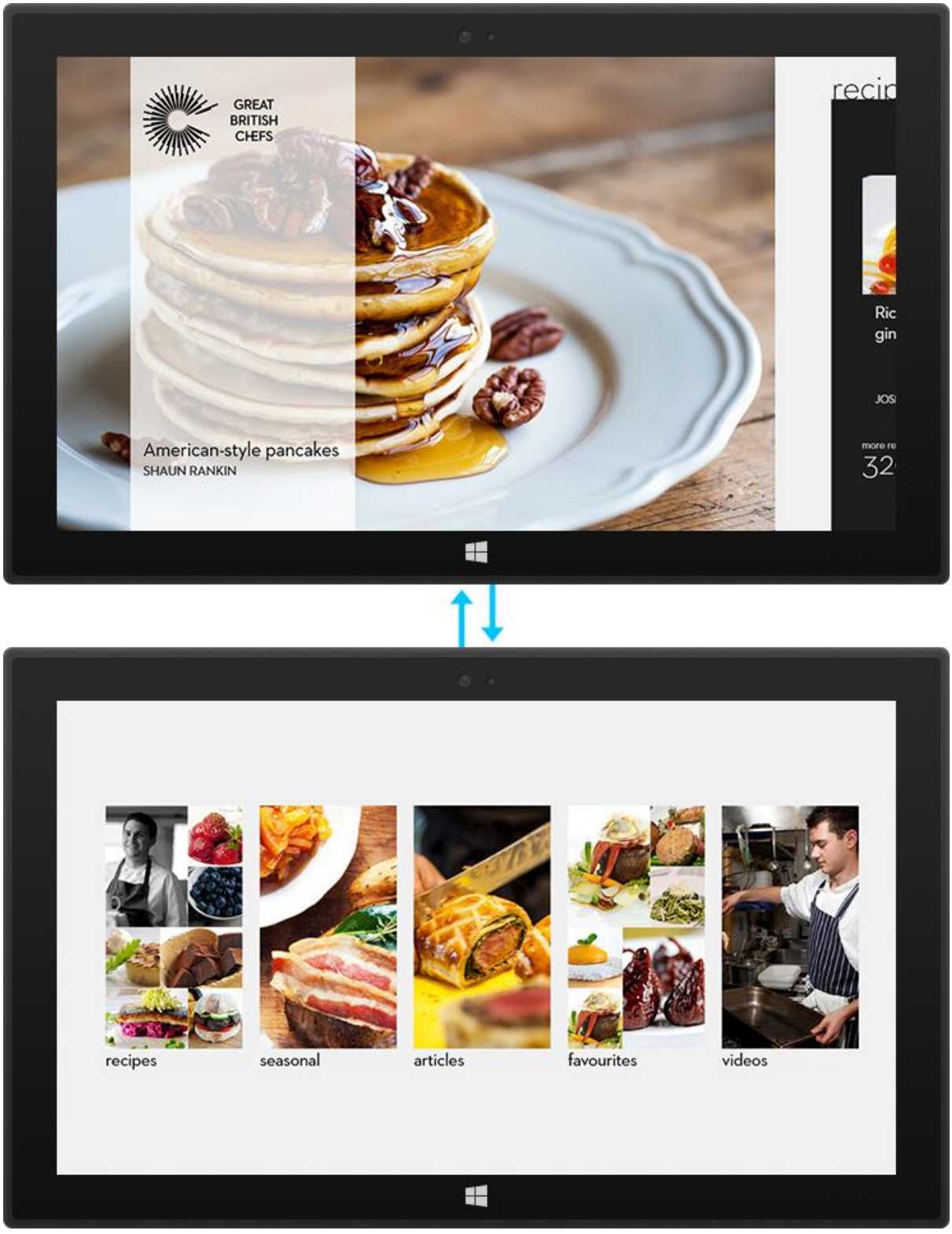


アプリ バーを創造的に利用して、デザインを向上させることをお勧めします。Fresh Paint では、上側のアプリ バーは専用のナビゲーション機能を持つだけでなく、ペイント ツール ボックスとしての機能も果たしています。

セマンティック ズーム

セマンティック ズームを使うと、 特にビューが長いリストの場合に、 軽快かつ柔軟にビューをスキャンして 移動することができます。

たとえば、Great British Chefs では、ハブ ページで 5 つのスポットライト セクションの後、ビジュアルで多彩な水平方向のパン セクションを置いています。このアプリでは、5 つの各セクションに簡単にジャンプするためにセマンティック ズームを使っています。



ナビゲーション、向き、ジェスチャ (Windows Phone ストア アプリ)

Windows Phone アプリは、ユーザーがリンクを使ってさまざまなコンテンツのビュー間を移動し、ハードウェアの戻るボタンを使って戻ることができるモデルを基盤にしています。このモデルの目的は、Windows Phone のページ ナビゲーション モデルに自然になじむビュー ベースのアプリの作成を簡単にすることです。

ページとフレーム

ページ ナビゲーション モデルは、ハブとスポークのシステムです。つまり、アプリ内の他のページへのリンクを明示的に追加しない限り、ユーザーは前に見たページを表示するには戻るボタンを使う必要があるということです。これは、Web ブラウザーで Web ページの履歴を表示して移動するしくみに似ています。

システムでは、ユーザーがアクセスしたページを追跡し、各ページをバック スタックと呼ばれる場所に配置します。そのため、ユーザーが戻るボタンを押すと、バック スタックに最後に保存されたページが表示されます。バック スタックに配置できるページ数に制限はありません。

たとえば、ユーザーがページ 1 (p1)、ページ 2 (p2)、p1、p2、ページ 3 (p3)、p1 の順に移動すると、p1、p2、p1、p2、p3、p1 というバック スタックが作られます。ユーザーがバック スタックにある 2 つ目の p2 インスタンスのコンテンツに変更を加えた後で戻るボタンを使ってバック スタックにある 1 つ目の p2 インスタンスに戻った場合、ページのデータが更新されていない限り、先ほどの変更は表示されません。表示されるのは、ナビゲーションにおけるその時点でユーザーに表示された状態のページのスナップショットです。

重要な注意: ユーザーのアプリ ナビゲーションに影響を与える可能性のあるページ間リンクやボタンを実装するかどうか、入力時にページを更新するかどうかは慎重に検討してください。

次に、Windows Phone のアプリ ページ モデルのコンテキストにおける定義を示します。

- ページ: ユーザーが認識できる永続的な状態のコレクション。情報、記録可能なコンテンツ、または他のページへのリンクを含むページとして表示できます。

- 画面: ポップアップ ウィンドウ、ダイアログ ボックス、スプラッシュ スクリーンなど、記録可能なコンテンツを含まない一般的な UI 画面。ユーザーが認識できる永続的な状態のコレクションではありません。

次の図に、複数のページや画面で構成される架空のアプリ構造を示します。

ページについて

アプリは以下のページで構成されます。

- ホーム ページ
- ウィジェット ページ
- ウィジェットの詳細ページ (ピボットを含む)
- ガジェットの詳細ページ
- 検索ページ
- 設定ページ

スプラッシュ画面やログイン画面はページではありません。ページは、ユーザーが認識できる永続的な状態のコレクションと定義されますが、スプラッシュ スクリーンはアプリが起動する前に表示される単なるプレースホルダーであるためです。スプラッシュ スクリーンについて詳しくは、「基本的なグラフィックス、視覚的なインジケーター、通知 (Windows Phone ストア アプリ)」をご覧ください。

また、ログイン画面では、資格情報の入力を求めるだけのため、状態データは含まれません (最後に入力したユーザー名を記憶する場合は例外です)。

重要な注意: Windows Phone のすべてのビューにページが必要なわけではありません。

UI をページにするかどうかを決定するときには、次のような点を検討してください。

- ユーザーが明示的にそのページを閲覧したいか。
- ユーザーはそのページを表示したことを覚えていてまたそこに戻りたいと思うか。

ページは、アプリ内で独立したコンテンツのセクションを保持し、個別の画面としてユーザーに表示されます。アプリ内でコンテンツを表示するさまざまなページを必要なだけ作成して UI を構成し、必要に応じてフレームやページからそれらのページへのナビゲーションを提供できます。1 ページしか必要としない単純なアプリもあれば、多数のページを必要とする複雑なアプリもあります。

全画面ビュー

全画面ビューの実装で、ステータス バーや下部のアプリ バーをオプションで表示することもできます。ただし、全画面ビューではこれらのバーは既定では表示されないため、可視性のプロパティを使って明示的に定義する必要があります。全画面ビューで一番望ましいのは、ユーザーがコンテンツ エクスペリエンスに集中できるようどちらのバーも表示しないことです。ステータス バーまたはアプリ バー (またはその両方) が表示されていない全画面モードでも、通知や電話の着信は表示されます。ステータス バーとアプリ バーについて詳しくは、「基本的なグラフィックス、視覚的なインジケーター、通知 (Windows Phone ストア アプリ)」をご覧ください。

注 全画面ビューの設定を使うと、ユーザーが時計にアクセスできなくなります。コンテンツで必要な場合を除き、全画面ビューを使わないことをお勧めします。

ページ間のナビゲーション

Windows Phone でのナビゲーションは、ページ間の遷移として定義できます。

ただし、スプラッシュ スクリーンからホーム ページへの移動など、本当の意味でナビゲーションと見なされない遷移もあります。ページとは見なされない場所からの移動は遷移と呼ばれます。

次のセクションでは、効果的なページ モデル使用のベスト プラクティスを紹介します。

画面と非ナビゲーション遷移

ログイン画面などの一時的な UI には、完全なナビゲーションを必要とする個別の画面を実装するのでなく、ポップアップ コントロールを使って、画面の一部分だけを含むコンテンツを表示することができます。ユーザーが戻るボタンを使ってダイアログ ボックスを閉じられるようにするには、コードに **BackKeyPress** イベントを追加し、ポップアップ ウィンドウの表示中に **e.Cancel** を **true** に設定します。

複数のコンテンツ ビュー

複数のコンテンツ セクションを表示するページでは、ナビゲーションを使わなくても、ページ上のコントロールを新しい DataContext に再バインドすることでコンテンツ間を移動できます。また、ページ内で複数の UserControl インスタンスを読み込むか、新しいコンテンツを表示するその他のメカニズムを使って再バインドすることもできます。ユーザーがどのようにして項目間を前後に移動するかは開発者が決定できます。たとえば、アプリ バーの [戻る] ボタンと [次へ] ボタンを使うことができます。ただし、ローカルの遷移では [戻る] ボタンを多用しないことをお勧めします。

状態の保存と廃棄

アプリが廃棄された場合にユーザーが手順をたどれるように、特定のページで発生した遷移の履歴を保存しておくことができます。前や次を閲覧する単純なシナリオの場合は、ページの状態を保存するだけです。これを Frame クラスと共に実行することで、廃棄状態から戻ったときにデータセットを走査するために必要な情報がすべて得られます。リンクされた項目間を自由に閲覧できるなど、複雑なローカルの遷移履歴が発生するアプリの場合は、履歴の一部をページの状態に保存することもできますが、保存する項目を妥当な数に限定する必要があります。重要な点は、ユーザーがハードウェアの戻るボタンを使って、前に表示した項目ではなく前のページに戻るようにすることです。

次の表に、アプリの各部分がページと見なされるかどうかを示します。

画面の種類	ページ	説明
スプラッシュ スクリーン	×	これは起動時のエクスペリエンスの一時的な部分であり、ユーザーがこの画面に移動することはできません。
ハブ エクスペリエンス	○	Windows Phone アプリの一般的なホーム画面のアプローチ。
詳細ページ	○	このページは、クエリ文字列でパラメーター化されたデータ中心のアプリに一般的です。
ピボット項目	×	ピボット項目とは、コンテンツ内でユーザーにとって重要な部分となるピボット コントロールの小さなコンポーネントです。
ログイン、または エラーダイアログ	×	これはアプリの状態によってトリガーされる一時的な UI であり、ユーザーがこの画面に直接移動することはありません。
項目の列挙	×	これはナビゲーション手段としてではなく、インプレース アクティビティとして類似コンテンツを閲覧するために使われます。

次の表に、さまざまな種類の UI 実装の処理に使用できる方法をまとめます。

UI の種類	実装	戻るボタンの動作	廃棄の動作
ページ	Page	戻るボタンは、自動的にアプリを前の状態に戻したり、終了したりします。データが失われた場合以外は、オーバーライドしないでください。	自動的にバック スタックに残ります。
画面の一時的な UI	ポップアップ または 子ウィンドウ	ポップアップ ウィンドウを取り消すには、アプリでオーバーライドする必要があります。スクリーン キーボードとメッセージ ダイアログは、戻るボタンが押されると自動的に取り消されます。	アプリでは、ナビゲーション時にポップアップ ウィンドウを閉じるか、取り消す必要があります。

項目の列挙	UserControl	該当なし: 親ページの内部でホストされます。	アプリは、必要に応じてアクティブな項目を保存する必要があります。

向きと軸について

Windows Phone プラットフォームのユーザーは、特にビデオ、地図、ゲームなどの要素を表示するアプリでは、縦横両方の向きがサポートされることを期待します。向きについて詳しくは、「初めての Windows Phone」の「画面の向き」をご覧ください。

また、ユーザーは、ナビゲーションが X 軸 Y 軸両方向に働き、上下だけでなく電話画面全体にスワイプすることで新しいコンテンツを発見できることを期待します。アプリで両方の向きと軸を使う方法を計画してください。

横向きで実行できるアプリを設計する際は次の点に注意してください。

- 縦モードでも使用できるようにするか、1 つの向きに固定するかを決定します。
- 横向きは横長の画面スペースを最大限に使う場合に便利です。ユーザーが電話の向きを変えたときに、より多くのコンテンツを画面に表示できるようになったことを強く印象付けるため、コントロールやレイアウトを再配置することができます。
- 横向きは、メディア コンテンツの一般的な縦横比であるため、その表示によく使われます。アプリを横モードにする主な目的がコンテンツ表示である場合は、コントロールやナビゲーションをフェード アウトさせて、表示エクスペリエンスを向上します。

タッチ ジェスチャ

タッチ ジェスチャは、ユーザーが Windows Phone の UI を操作するための主な手段です。Windows Phone のタッチ ジェスチャには、タッチ スクリーン上を 1 本または複数の指で操作するタップやスワイプがあります。

Windows Phone SDK で提供されているコントロールは、タッチ操作要素として使われます。すべてのタッチ操作要素は、常にタッチ操作に適切なサイズにする必要があります。タ

ッチ ターゲットについて詳しくは、「Windows Phone の対話操作と操作性」をご覧ください。

Windows Phone では、次のシングル タッチ ジェスチャとマルチタッチ ジェスチャがサポートされています。

シングル タッチ

- タップ
- ダブルタップ
- パン
- フリック
- 長押し

マルチタッチ

- ピンチとストレッチ

タップ

タップとは、画面の特定の領域にすばやく 1 回触れて離すことです。

タップ ジェスチャに関連付けられている動作は 2 つです。

- 指で押すことでタッチしたことを示す
- 指を離すことでアクションを実行する

また、動作の進行中にタップするとその動きの慣性が完全に止まります。

ダブルタップ

ダブルタップとは、特定の領域内をすばやく 2 回タップすることです。

ダブルタップすると、コントロールまたはアプリのズームの拡大/縮小の状態が切り替わります。アプリは現在のズームの状態を判断し、それに応じて拡大または縮小を行います。アプリでは拡大/縮小の状態が定義されます。

パン

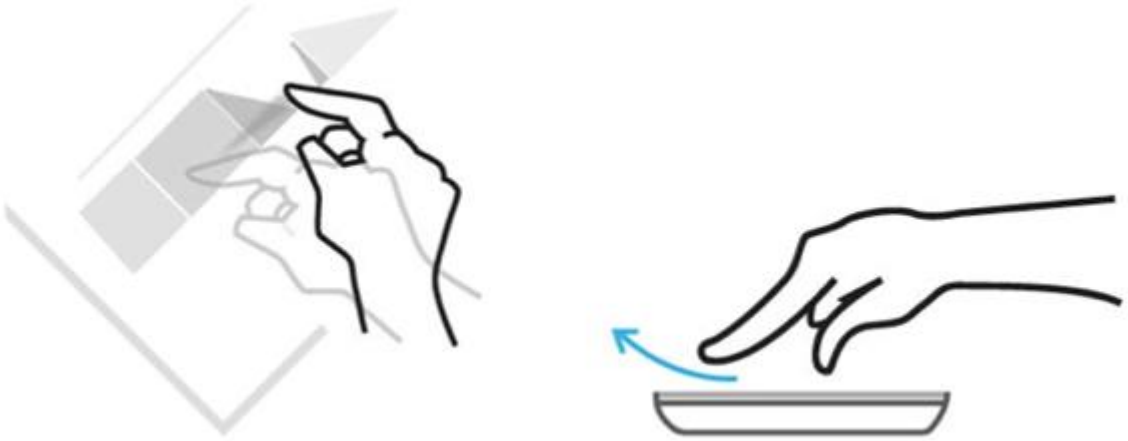
パンとは、1 本の指を画面に置いて任意の方向に動かすことです。指を画面から離すとパンジェスチャが終了します。

パン ジェスチャに関連付けられている動作は 2 つです。

- コンテンツを直接操作して移動することができます。コンテンツは指の動きを追ってそのとおりに動きます。コントロールやアプリでは、サポートするパン方向を決めることができます。この動作には、水平、垂直、またはその他の方向を指定できます。コンテンツが中間的な状態に移動された場合、最も近い状態にスナップする必要があります。
- 特定の項目を移動したり並べ替えたりすることができます。項目は指の動きを追って移動し、画面から指を離すとその場所にドロップされます。

フリック

フリックとは、1 本の指で画面に触れて、任意の方向にすばやく動かしながら画面から離すことです。パン ジェスチャの後にフリックすることもできます。



フリック ジェスチャでは、スクロール可能な領域のパンやオブジェクトの移動が慣性付きで行われます。コントロールやアプリは、特定のフリック方向をサポートするように構成できます。この動作には、水平、垂直、または別の方向を指定できます。水平または垂直を指定した場合、その他の方向への動きは、垂直または水平の動きに変換されます。

通常、フリックはキャンバス全体を移動しますが、個別のオブジェクトを移動するように指定することもできます。

ピンチとストレッチ

ピンチとストレッチとは、2 本の指で画面上の特定の個別の領域に触れ、2 本の指を近づける (ピンチ) か離す (ストレッチ) ことです。

ピンチとストレッチを行うと、2 本の指の中間地点を中心としてコンテンツが連続的に縮小/拡大されます。ピンチまたはストレッチがフリックで終了する場合、慣性付きで拡大/縮小が行われるようにアプリを反応させることができます。

長押し

長押しとは、1 本の指で特定の領域に一定時間触れることです。通常、長押しジェスチャは、項目のコンテキスト メニューやオプション ページを表示するために使われます。

マルチタッチ ポイント

Windows Phone では、一意のアプリ操作を実現するために、少なくとも 4 つの同時ユーザー タッチ入力ポイントをサポートしています。ゲームや楽器のアプリがその簡単な例です。

タッチの端が 3.5 mm 以上離れていてすべてのジェスチャがサポートされている場合に、画面に対する直径 7 mm 以上のタッチが一意のタッチとして認識されます。

タッチ ポイントを増やすとプロセッサの負荷も増加します。そのため、3 つ以上の同時タッチ入力ポイントを実装する場合はパフォーマンスへの影響に注意してください。
Windows Phone では最大 10 個のタッチ ポイントをサポートしていますが、ハードウェア画面によっては 4 つまでしかサポートしていない場合もあります。

ナビゲーション パターン (Windows Phone ストア アプリ)

たとえばバンキング アプリの場合、残高の確認などの情報がユーザーに表示されます。アプリがユーザーから収集する入力は、口座間の残高移動のようなものになる場合があります。その一方で、レース ゲームでは車と道路が表示されます。ユーザーからの入力は、車を操縦するために使われます。

有効な操作モデルを実現するには、情報のデザインと操作方法に多くの時間を費やす必要があります。ナビゲーション モデルは、それぞれの画面に何が表示され、ある画面から別の画面にどのように移動するかを決定します。「最良の Windows Phone アプリを設計する」では、画面のマップと画面間の接続について概略を練るための方法について説明しています。すべてのアプリは、ユーザーが最良のエクスペリエンスを得られる操作モデルに従う必要があります。

このセクションでは、Windows Phone アプリを設計する際に使用する可能性がある、アプリの操作モデルの主要なカテゴリについて説明します。これらのスタイルは、情報が表示および取得される方法や、ユーザーがアプリのさまざまな領域をナビゲートする方法を決定します。各スタイルは、ユーザーの操作エクスペリエンスをできるだけ楽しくするための特定

の要件を満たしています。一部のスタイルは他のスタイルと組み合わせて使うことができます。その場合はその旨を示し、例を紹介します。

上位レベルのアプリ操作モデルが多くあり、好きなものを選択できます。必要と考えられるモデルの感触を得るために、それぞれの概要を読むことができます。

アプリのビュー管理

ユーザーが多くの画面の間を移動する必要がある場合は、UI をすっきりと表示する方法が必要です。ユーザーに表示するビューが多くなるほど、アプリのビュー管理はより重要になります。このセクションの最初の数トピックは、より複雑なシナリオの操作のモデルを示しています。それ以降のトピックは、よりシンプルな単一ビューのアプリまたはシンプルなリストを必要とするシナリオに適したモデルについて説明します。

最初に行う必要があるのは、目的とするアプリのスタイルを選ぶことです。たとえば、20 個の最も一般的なアメリカ手話を学べるシンプルなフラッシュ カード アプリを開発している場合は、「Windows Phone 用の均一化されたページ シャッフル」だけを読めばよいでしょう。

アプリのスタイル

各操作タイプについて説明します。一部はアプリの全体的なナビゲーション デザインとして使用されるため、UI レイアウトのために下位レベルの操作のデザインが必要です。既に説明したように、一部の種類はそれ自身によって使用され、完成したアプリとなります。また、それらの種類をいくつか組み合わせてアプリを作成することもできます。たとえば、アプリ全体のナビゲーションの大まかなスタイルを選択した後、UI のサブ領域で UI 表示用に別のスタイルを選択できます。

- Windows Phone のホーム ページ メニュー (Hub コントロールまたは Pivot コントロール) を持つセントラル アプリ ハブ
  アプリのメイン ページは、メイン メニューからアプリのさまざまな領域に入るためのスタート サイトとして使用されます。
- Windows Phone のパネル領域 (Hub コントロール) を持つセントラル アプリ ハブ

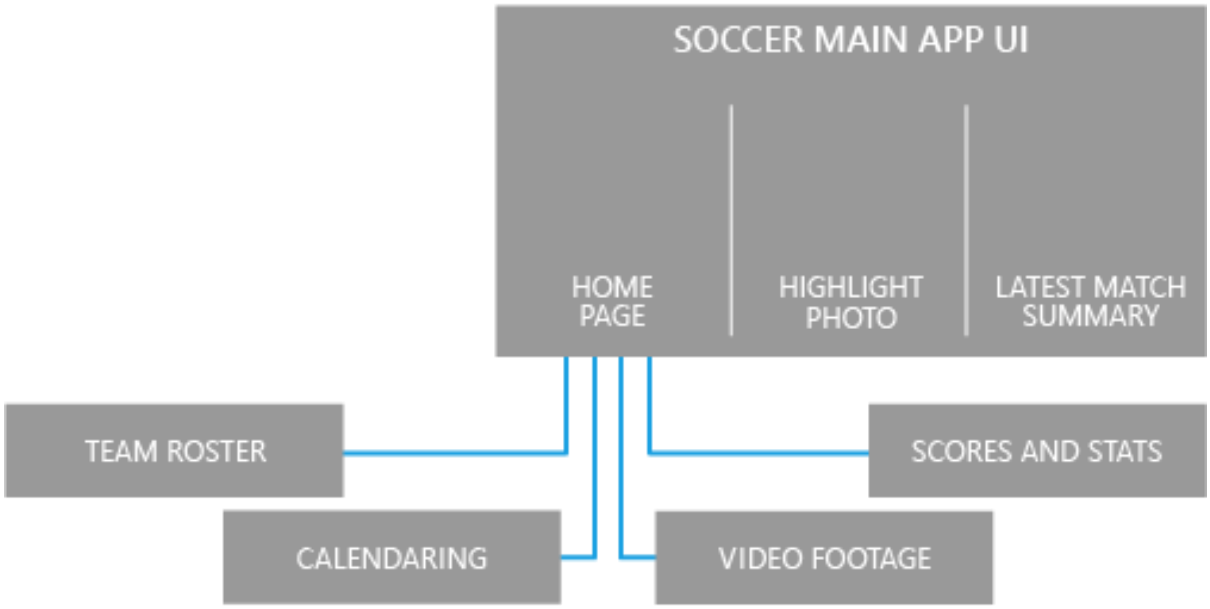
アプリのメイン ページは、すべて同じレベルのアプリのさまざまな領域に入るためのスタート サイトとして使用されます。

- Windows Phone のアプリ タブ (Pivot コントロール)
  フィルター処理された一意のビューを表すタブを持つ単一のテーマに基づいてユーザーがタブ移動できる、いくつかの領域で構成されるアプリ。
- Windows Phone 用の詳細ドリルダウンを備えたリスト
  詳しい情報を表示するためにユーザーがタップする項目の一覧を表示する必要があるアプリ。
- Windows Phone 用の均一化されたページ シャッフル
  トランプのように、フリックでめくることができる均一化されたシンプルなページ。

ホームページ メニューを持つセントラル アプリ ハブ

多くの機能を備えたアプリを設計しているとします。機能について考えた結果、明確な領域にそれらの機能を整理できと判断しました。それらの領域は、最終的には、ユーザーがアクセスできるアプリの別個の部分になります。ユーザーがこれらの UI 領域を移動するための簡単な方法を設計する必要があります。このようなアプリには、ユーザーがアプリの各サブ領域に移動する際に使うセントラル アプリケーション ハブが必要です。

たとえば、サッカー チームを管理するためのアプリを設計するとします。このアプリには、いくつかの機能領域 (試合と練習のスケジュールを作成する領域、チーム名簿情報の領域、ストアと選手の統計の領域、さらに過去の試合のビデオ映像に使う領域) が必要です。これらの各 UI 領域に移動する手段をユーザーに提供します。これは、アプリの起動時に表示されるホーム ページを持つセントラル アプリケーション ハブを実装して実現できます。



サッカー チーム管理用のセントラル アプリ ハブ UI

ユーザーは、セントラル UI ハブからアプリケーションの任意の領域に移動できます。サブ領域に移動したら、その時点で必要な UI を表示します。たとえば、サッカー チーム アプリでは、ユーザーはセントラル アプリケーション ハブ ページから、特定の試合の選手の統計情報を確認できるページに移動できます。ユーザーがアプリの特定のサブ領域で作業を終えて、別のサブ領域に移動する場合、まずセントラル アプリケーション ハブに戻ります。

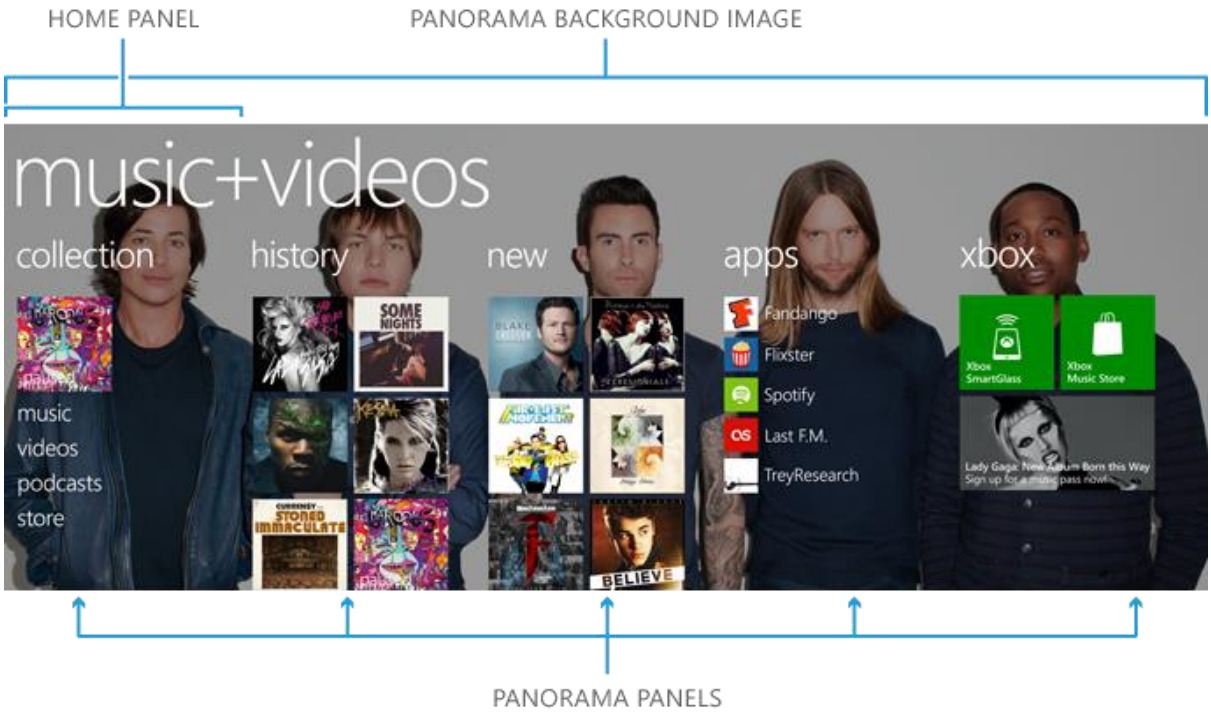
ユーザーは、[Scores and Stats] から Calendaring に直接移動することはできません。まずセントラル アプリケーション ハブに戻る必要があります。

注 アプリのさまざまな領域に移動するためのホーム ページを持つセントラル アプリケーション ハブを使うことは、ユーザーがアプリを効率的に操作できる優れた設計です。

アプリケーション ハブをユーザーに視覚的に表示する方法はいくつかあります。ハブ コントロールを使うことをお勧めします。このトピックではこの設計について説明した後、検討できるバリエーションについても示します。

ハブ コントロールをアプリケーション ハブとして使う

Microsoft では、セントラル アプリケーション ハブとして使うことができる、ハブ コントロールという UI コントロールを用意しています。このコントロールを使うと、ユーザーはアプリですべての機能領域に移動することができます。ハブ コントロールは、コントロール全体の背後に背景として表示されるハブ背景画像、アプリの起動時に表示されるホーム セクション、アプリの最上位に UI をセグメント化する追加セクション、という構造になっています。



ハブ コントロールは、画面の境界を超えて横方向へ拡大する、幅の広い仮想キャンバスをユーザーに表示します。ユーザーは、クリックすることでキャンバスを横方向にセクションごとに移動できます。アプリを起動すると、ハブの最初のセクションが表示されます。ここからは、希望するナビゲーションの動作に応じて使用方法にバリエーションがあります。各バリエーションについて説明します。

ホーム セクション

一番左にあるこの最初のセクションは、アプリのサブ領域に移動するための出発点であるホームとして使うことができます。このセクションには、ユーザーが選ぶことができる領域のメニューが表示されます。前の図に示した Music ハブでは、[ラジオ] を指でタップすると、実際には Music ハブ内のサブアプリケーションであるラジオ ページに移動します。ハブ UI は完全に置き換えられ、ユーザーには [ラジオ] の UI が表示されます。ユーザーが [ポッドキャスト] 領域に移動する場合、ハードウェアの戻るボタンを使ってハブのホーム セクションに戻り、メニュー リストで [ポッドキャスト] エントリをタップします。サブ領域の一覧がディスプレイより長い場合は、ホーム画面でスクロール ビューを使うことができます。

Music ハブのラジオ ページ

ホーム セクションにある移動先の一覧には、完全に新しいアプリに移動できるエントリが含まれていることがあります。たとえば、Music ハブのホーム セクションには、ストアに

移動するエントリがあります。これをタップすると、それ自体がハブ コントロールであるストアの音楽サブ領域に移動します。

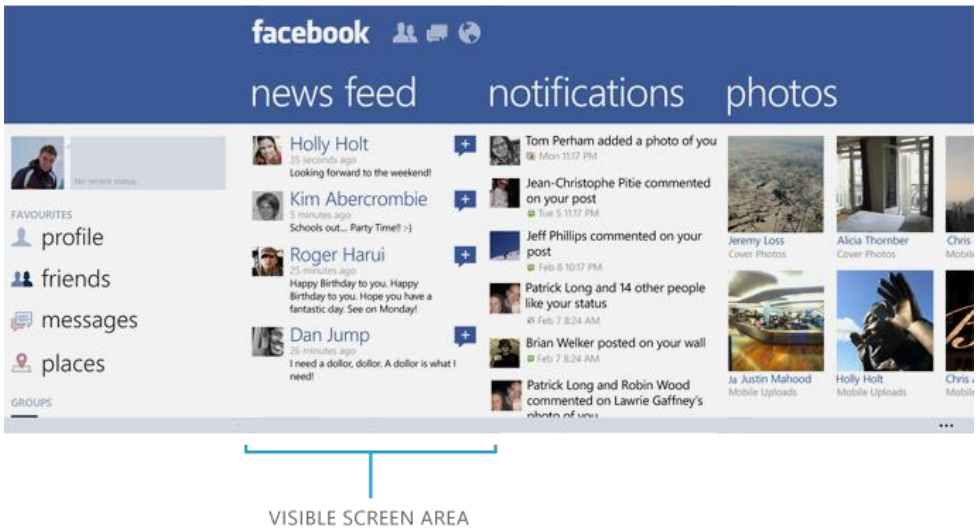
右側のセクション

ホーム パネルの右側のセクションには、ユーザーが簡単にアクセスできるようにする UI が含まれています。使い方は自由です。サブ領域にさまざまな方法でアクセスできるようにして、ユーザーを混乱させないでください。代わりに、右側でこれらのセクションを使って、サブ領域に含まれている内容に関するある種の概要情報を含めます。たとえば、Music ハブの場合、最近アクセスしたメディアを示す 2 つの追加のセクションがあります。

Music ハブの [履歴] セクションは、サブ領域まで移動する必要がなく、最近視聴したものにすばやくアクセスできるため便利です。[音楽] のサブ領域に移動して特定の曲を聴いた場合、その曲が [履歴] セクションに表示されてすばやくアクセスできるようになります。したがって、この場合、サブ領域でアクセスしたコンテンツにセクションからすばやくアクセスできます。

Facebook アプリは、ハブ コントロールを使ってユーザーが機能を簡単に移動できるようにしているアプリの別の例です。このアプリにも、ユーザーが移動できる領域の一覧が表示される、同じホーム セクションがあります。たとえば、friends エントリをタップすると、友人の一覧を管理する UI に移動します。

最上位のハブ コントロールでは、最上位に表示すると最も役に立つとデザイナーが考えたビューにすばやくアクセスできるようにするため、基本的には右側のセクションが使われます。ハブの most recent セクションは、実際にはサブ領域の UI です。ホーム セクションから移動するには、news feed をタップすると、そのサブ領域に移動します (このサブ領域は実際にはピボット コントロールです)。news feed サブ領域のピボット コントロールには、most recent、photos、links、videos の各項目があります。デザイナーは news feed まで移動した後 most recent に移動することをユーザーに求める代わりに、上部のハブ コントロールのセクションとして表示することにしました。同様のことは、上部のハブ コントロールにある photos セクションと events セクションにも適用されています。

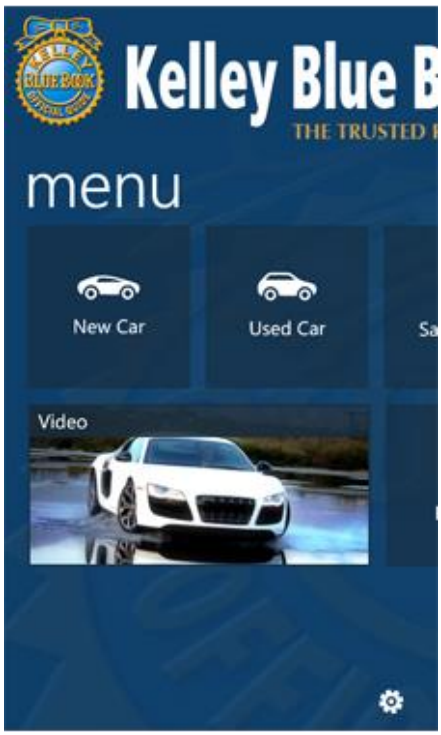


画面グリッドを使う

ホーム セクションから移動するための領域の一覧を表示する代わりに、選択可能な画像のグリッドを表示することができます。行うことは同じです。外見が少し異なるだけです。画像を使うと、その選択による移動先を判断しやすい視覚的なアイコンを表示できます。各画像の上にはテキストを重ねて、そのアプリ領域とアクティビティに関する詳しい情報を示すことができます。

3 x 3 のグリッドがある場合、必要に応じて最大 9 枚の画像を配置できます。この画像グリッドをホストするホーム セクションで、ユーザーは任意の画像をクリックして、アプリに用意されたさまざまな機能領域に移動できます。

このホーム セクションは、実際には複数の表示可能なセクション幅にまたがる領域です。たとえば、必要であれば、画像のホーム ページを右側に拡大して、より多くのスペースを占有させることができます。Kelley Blue Book アプリは、次の図に示すようにグリッドが右に伸びています。画像の残りの部分を表示するには、フリックして横方向にパンします。この場合のハブ コントロールのホーム セクションは、幅が 2 倍です。



Kelley Blue Book アプリのアプリケーション ハブ UI

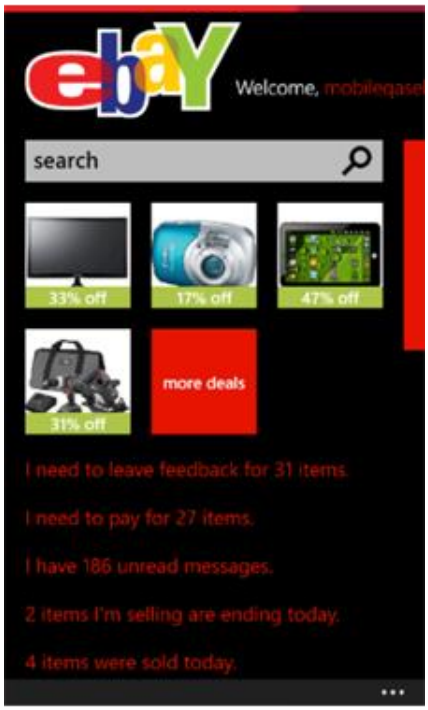
縦方向にスクロールする画像のグリッドを作らないでください。ユーザーにとってかなりわかりにくくなります。

ホーム セクションから開始しない

ハブ コントロールを使って、アプリの機能領域に移動するメニューやハブのように機能するホーム セクションを設置した場合でも、このセクションをアプリの起動時の最初のランディング ポイントとする必要はありません。他のセクションを最初に表示する理由は、効果を高めることだけです。たとえば、映画と上映時間が表示されるアプリがあり、ホーム セクションにはアプリの特定のサブ領域の一覧が表示されるとします。あまり関心のない内容をユーザーに表示する代わりに、最新の大人気映画のグラフィックが表示するセクションを配置できます。このようにして、より目を引く内容をユーザーに表示できます。

eBay アプリを開いても、最初にホーム セクションが表示されるわけではありません。この設計では、代わりに現在注目の取引を含むセクションが最初に表示されます。右にフリックした場合、クリックするとアプリのサブ領域 (Watching、Selling、Buying、Messages) に移動する画像が表示されるアプリケーション ハブ ホーム セクションに移動します。上部に検

索テキスト ボックスが使われている点にも注目してください。これは、すばやく検索を行うことができるように上部に配置すると役に立つ UI 要素です。

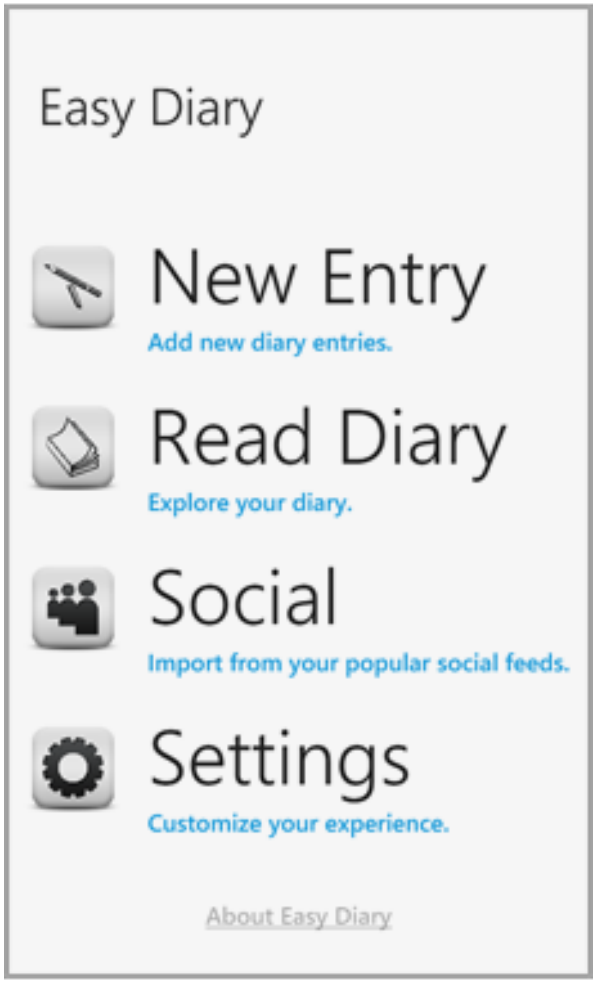


eBay アプリのアプリケーション ハブ UI

縦方向にスクロールする画像のグリッドを作らないでください。ユーザーにとってかなりわかりにくくなります。

UI 機能領域のカスタマイズされたメニュー

Microsoft によって用意されたハブ コントロールを使う以外の方法もあります。ハブを使うと、最上位に複数のセクションを表示できます。ただし、これらのセクションが必要でない場合もあります。次の図に示すように、代わりに簡単な一覧を 1 つのページに表示することができます。これは、アプリが開いたときにユーザーに表示されるページです。このアプリでは、移動先の各サブ領域のテキスト タイトルと説明の左側でアイコンが使われています。



easy diary アプリのアプリケーション ハブ UI

移動可能なサブ領域が芸術的な独自の方法で表示されたグラフィック背景を使って、独特な操作エクスペリエンスを表示することを決定できます。これは、自身のブランドを確立するために行うことができます。一部の種類のエンターテインメント アプリには、さらに当てはまります。魅力的な外見のホーム ページの例を次に示します。

MyComic アプリ

同様のアプリを 1 つにまとめる

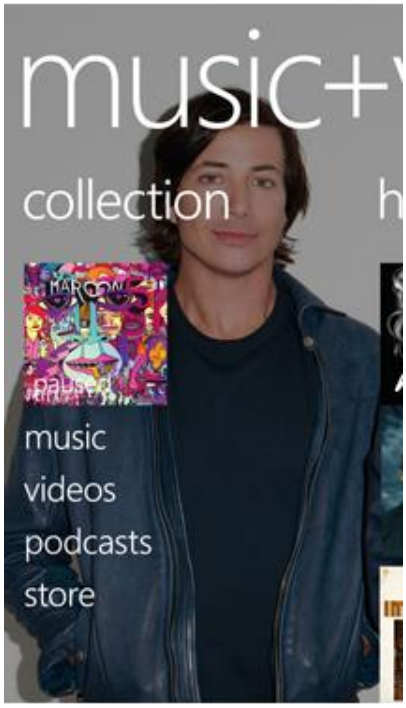
既に説明したように、アプリに明確な機能領域が多く存在しているため、それらの機能をそれ自体は別個のアプリに分割することを検討したかもしれません。そうする必要はありません。1 つのアプリのままにすることができます。アプリを開いたときに表示されるメイン画面は、1 つのアプリを構成する実際のサブ アプリケーションにアクセスするためのスタート地点として機能します。このメイン アプリケーション ハブ セクションは、ユーザーがアプリ アイコンをクリックしてアプリを起動すると表示されるページです。

独自の機能領域ごとに別個のアプリを作らないことをお勧めします。これで問題となるのは、ユーザーがアプリを終了してから別のアプリを起動する必要があるという点です。ユー

ザーが操作するアプリが 9 つあるとします。代わりに、ユーザーが作業を開始するための 1 つのアプリを作り、その 1 つのスタート地点からアプリの各領域にアクセスできるようにすることをお勧めします。アプリの設計者は、ユーザーがセントラル ハブに移動し、用意されているすべての機能が 1 つの中央の画面に表示されるようにしてください。

ナビゲーションのレベル

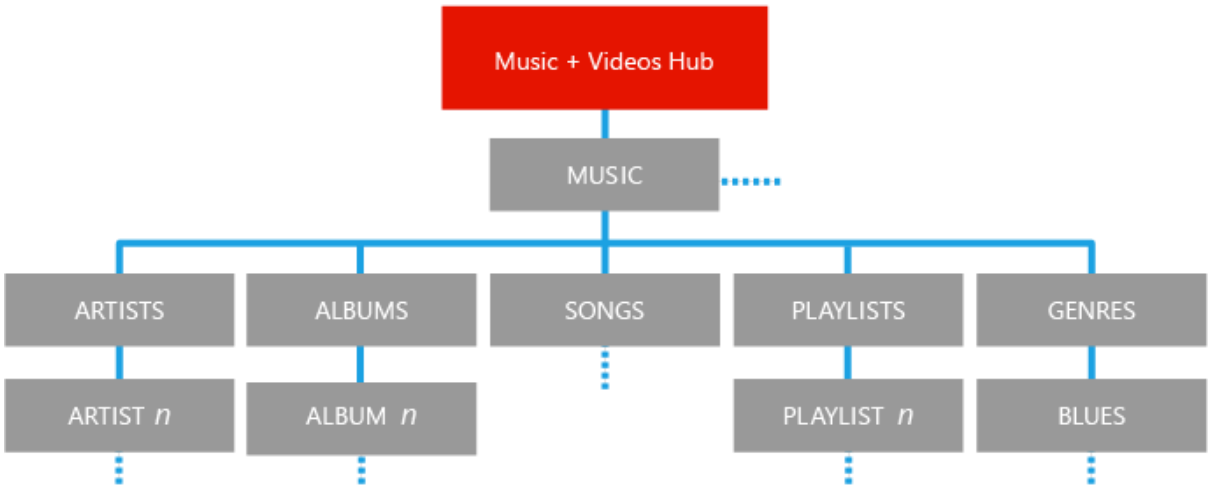
下すのが難しい決定は、アプリの機能をどのように分割し、どのナビゲーション レベルでユーザーが移動できるようにするかということです。ホーム セクションがあるハブ コントロールを使う場合は、ユーザーが一覧から選択肢をタップしたときに表示される UI を決定する必要があります。選択後、ハブ コントロールから、1 つのページ (または、場合によってはピボット コントロール) などの新しい UI コントロールに移動します。次に示すように、Music ハブのホーム セクションを使ってこれを示します。



Music のアプリケーション ハブ

Music ハブの場合、[ラジオ] を選ぶと、聴いているラジオ局を変更できる 1 つの UI ページが表示されます。このサブ領域では、さらに移動するためのリスト コントロール、ハブ コントロール、ピボット コントロールが使われていません。ただし、ホーム セクションから

[音楽] を選ぶと、所有している曲の別のビューが表示されます。次の図は、[音楽] の選択した場合に移動する UI ツリーの一部を示しています。

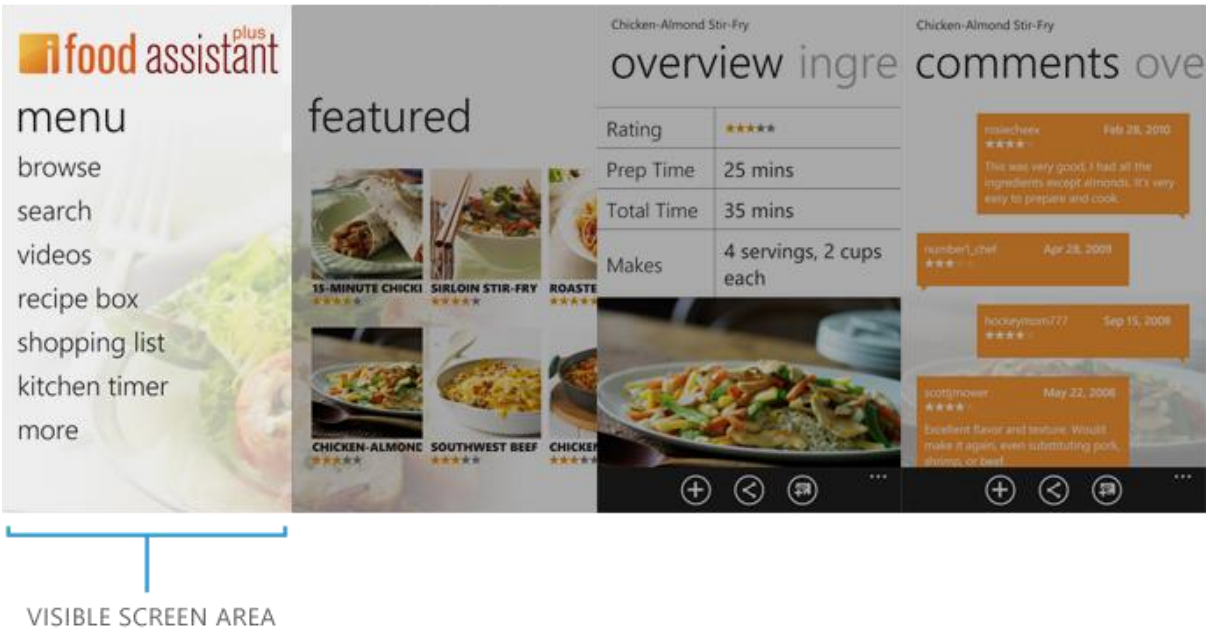


[音楽] 領域が表示されている場合、横方向にスクロールできるピボット コントロールが表示されます。[音楽] の下で最初に表示されるピボット ページは常に [アーティスト] ピボット項目です。そこから、さまざまなピボット項目を使って左右に移動できます。各ピボット項目を使うと、選択可能な曲をさまざまな方法で表示できます。

ユーザーがアプリ内の目的の領域に移動したら、その時点で必要な UI を表示します。アプリの特定のサブ領域から、ユーザーは実際の目的のタスクを行うことができます。この第 2 レベルでは、移動先領域の別の一覧を表示することは推奨されません。メイン ページのハブ コントロールと第 2 レベルの詳細のみ保持するのが最適です。この第 2 レベルからは、ユーザーはホーム セクションに戻ってからアプリの別の領域に移動します。戻るには、ハードウェアの戻るボタンを使います。

背景画像を使う

ハブ コントロールを使うと、すべてのセクションの背後に画像を表示できます。常に同じ画像を使うことも、プログラムにより画像をときどき変えてテーマを変更することもできます。アプリを使っているユーザーの興味に関連する画像を表示できます。あまり雑然としておらず、UI コンテンツと干渉することがない画像を使ってください。



複数のホーム パネル

移動先のサブ領域の一覧を含むセクションが複数必要な場合があります。ナビゲーション領域の 2 つの異なる一覧に分割する必要が生じることがあります。この場合、どちらもホーム セクションとして機能する 2 つのセクションが存在することになります。

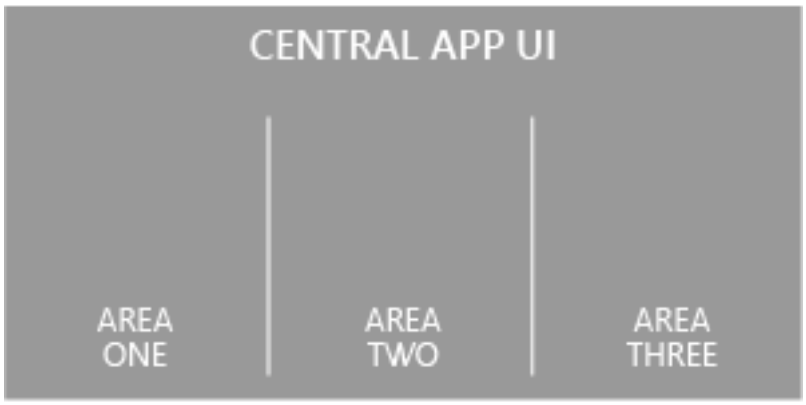
メイン起動画面

アプリの起動時、セントラル アプリ ハブに直接移動できないことがあります。1 つの理由は、アプリへのアクセスをユーザーに許可する前に、ログオンまたはパスワードによる何らかの種類のロック解除をユーザーに表示する必要がある場合です。アプリに入る前に、ブランド起動画面をユーザーに表示することもできます。

easy diary アプリのログイン UI

パネル セクションを持つセントラル アプリ ハブ

このトピックのモデルは、「ホーム ページ メニュー (ハブ コントロールまたはピボット コントロール) を持つセントラル アプリ ハブ (Windows Phone ストア アプリ)」をご覧ください。前の場合と同じように、ユーザーに表示される機能のさまざまな領域があります。ここでの違いは、ホーム ページ メニューがなくても、すべてのメイン UI を最上位に表示できることです。



サッカー チーム管理用のセントラル アプリ ハブ UI

移動先のサブ領域がアプリにない場合、横方向にアクセス可能なセクションの 1 つのセットにすべての UI を配置できます。その場合もハブ コントロールを使うことができますが、この場合は、ホーム セクションが必要ありません。これは、Windows Phone の People ハブにも当てはまります。次の図は、ハブの基本レイアウトは同じままであることを示しています。

People ハブでは、各セクションに固有の UI が存在し、機能の領域が表示されます。設計のこのモデルと「ホーム ページ メニュー (ハブ コントロールまたはピボット コントロール) を持つセントラル アプリ ハブ (Windows Phone ストア アプリ)」で説明されているモデルの違いは、アプリの下部領域のサーフェスをハブで再現するだけではないという点です。前のモデルでは、ホーム セクションの右側のセクションで、メニューのナビゲーションを通じてアクセスできるコンテンツをすばやく確認できます。ここで説明するモデルでは、他の場所にある領域のサーフェスがセクションに再現されません。

各領域のセクション

設計に関して、アプリの起動時に選ぶセクションを決定します。ユーザーが前回アプリを終了したときに表示されていたセクションを選ぶことができます。常に最初のセクションを選ぶこともできます。最初のセクションとは、背景画像のフラッシュが左に表示され、ハブ タイトルの左端がその上に表示されたセクションです。

右側に表示するセクションの数を決めることができますが、セクションは合計で 4 つ以下にすうことをお勧めします。セクションが 4 つ以上ある場合、ユーザーは自分のいる場所を把握しにくくなります。

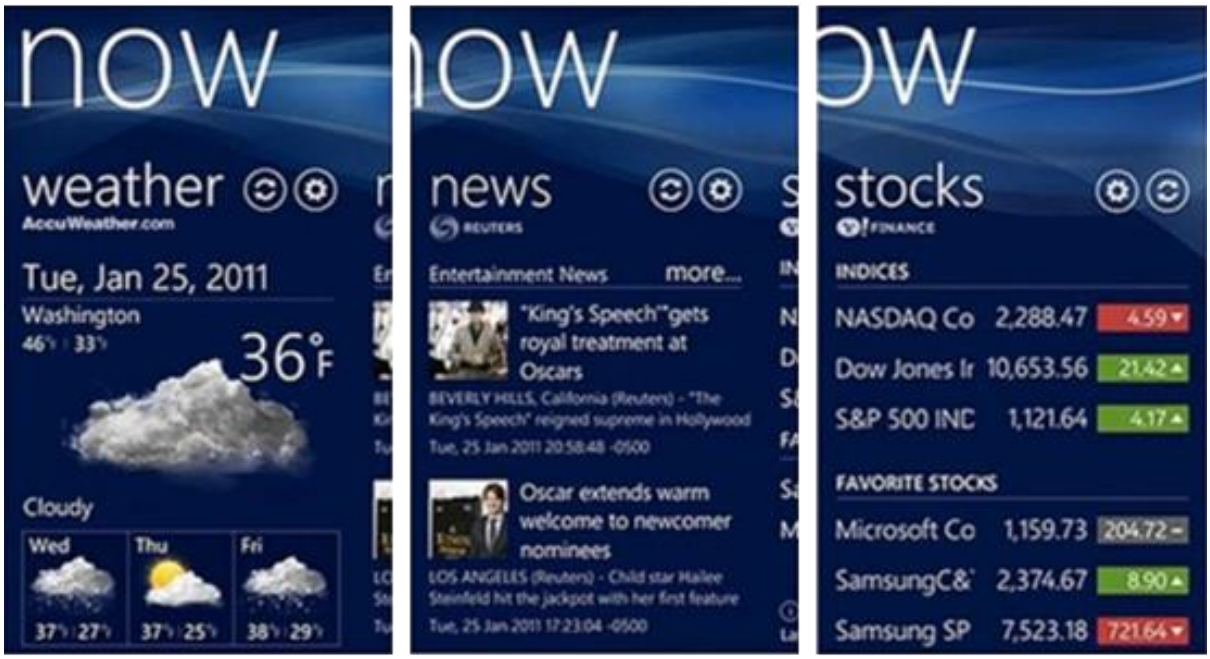
どの UI も各セクションに自由に配置できます。主な機能領域が 4 つある場合、1 つの領域を 4 つの各セクションに配置できます。最初のセクションは、ユーザーがアプリの起動時に最も見たいと思っていると考えられるコンテンツで設計できます。上の図は、セントラル アプリケーション ハブとして使う、この種類のハブの使用方法を示しています。

ハブの各セクションには、クリックするとその項目の詳細が表示される項目が表示されます。たとえば、People ハブには、タップ可能な項目が表示される [最新情報] セクションがあります。項目は、縦方向にスクロール可能な一覧に表示され、Facebook への投稿を表している場合があります。これらのいずれかをクリックすると、ハブから離れてその投稿の詳細が表示されます。[最新情報] セクションには、各エントリの複数のヒット ターゲットが表示されます。1 つのヒット ターゲットをタップすると投稿に移動し、投稿に対する返信が確認できます。他のヒット ターゲット、人物の名前です。これをタップすると、その人物のピボット コントロールに移動し、その人物に関する情報のピボット項目をいくつか確認できます。

別の例

各セクションには、独自の明確な領域が存在することがありますが、必ずしも他のセクションと関連していません。この場合も、ユーザーが個々のセクションのコンテンツをタップすると、メイン ハブから移動します。移動先は、ホーム セクション メニューからのように完全に新しいサブ領域ではなく、タップした内容の詳細です。

Samsung Now アプリには、ハブ コントロールをセントラル アプリ ハブとして使った 3 つのセクションがあります。このアプリのテーマは、朝や日中にすばやくアクセスする必要がある情報 (天気、ニュース、株式市場情報) をユーザーに表示することです。



Samsung Now アプリ

この場合も、ユーザーはハブから移動できます。たとえば、株式セクションの株式記号をクリックすると、その会社に関する詳細を含む 1 つのページに移動します。

アドバイス

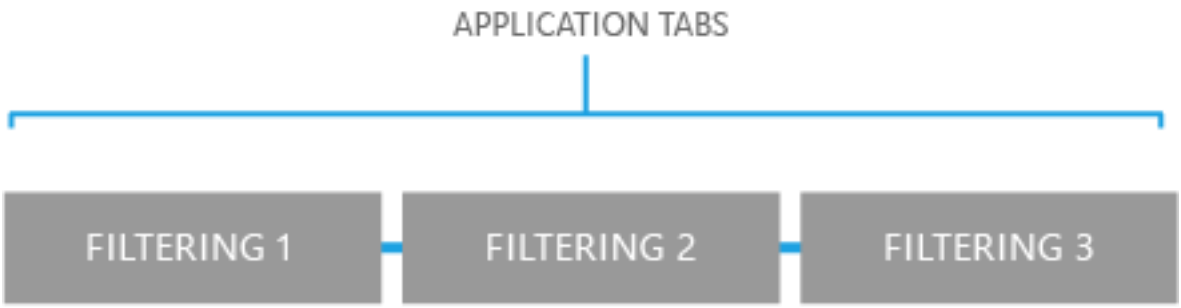
「ホーム ページ メニュー (ハブ コントロールまたはピボット コントロール) を持つセントラル アプリ ハブ (Windows Phone ストア アプリ)」でもこのトピックでもハブ コントロールを使っているため、同じアドバイスと注意事項の多くがここでも当てはまります。

アプリ タブ

アプリ タブ パターンは、ユーザーが頻繁に移動する UI の複数のページに使用されます。このパターンは、アプリが 1 つの中心的なテーマ (たとえば、映画や野球など) に基づいている場合は特に便利です。各ページには、アプリ全体で表示される総合的なデータに関連する内容が表示されます。アプリ タブ パターンは、アプリ全体を構成するために使われたり、アプリのサブ領域で使われることもあります。たとえば、アプリのメインの第 1 レベルとしてハブ コントロールを設置し (前のトピックで説明)、そこからアプリ タブ パターンを使うアプリのサブセクションにユーザーがナビゲートできるようにすることができます (ピボット コントロールを使用)。

自宅のコレクションにある映画を分類するアプリを作るとします。フィルター処理された映画のビューを複数表示するために、アプリ タブ パターンを使うことができます。フィルター処理された 1 つのビューには、お気に入りとマークされた映画を表示します。フィルター処理された別のビューは、アクション映画だけを表示します。フィルター表示された別のビューには、時間があるときに視聴しようと思っている映画を順番に表示します。もちろん、フィルター処理しないですべての映画のリストをページに表示することもできます。

次の図は、このアプリの構造を視覚化するのに役立ちます。アプリを起動すると、フィルター処理されたページの 1 つが表示されます。そこから、UI のタブのように機能する各ページを横にスクロールできます。



Outlook の受信トレイのアプリ タブ

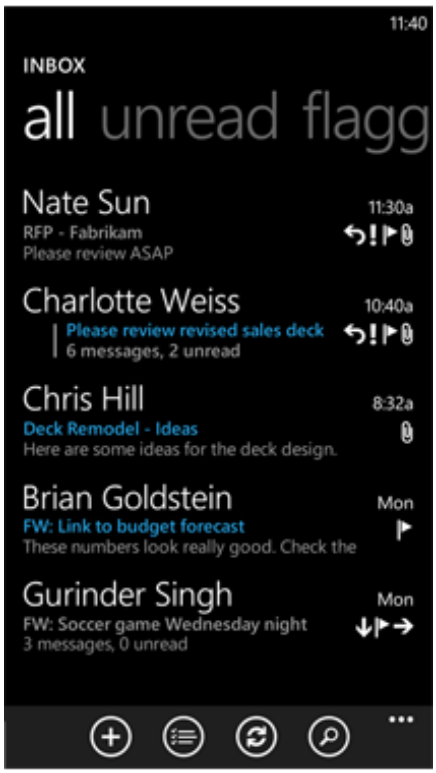
各タブには、ユーザーに表示するデータが、基本的には異なる方法でフィルター処理されて表示されます。フィルター処理されたそれらのビュー ページのいずれかが表示されているときにハードウェアの戻るボタンが押された亜売、アプリを終了します。

フィルター処理された異なるビューを表示する、1 つのデータの種類を持つアプリにはアプリ タブ パターンを使います。

ピボット コントロールをアプリ タブ実装として使う

ピボット コントロールを使うと、アプリ タブ パターンを実装できます。このコントロールでは、ユーザーが各ページ間を左右に移動できます (ピボット項目と呼ばれます)。

Microsoft の Outlook クライアント アプリでは、アプリに 1 種類のデータ (各ピボット項目に異なるフィルターが適用された電子メールの一覧) が表示されるため、アプリ タブ パターンが使われています。Outlook アプリで電子メールの受信トレイを見ると、3 つのピボット項目 (all、unread、urgent) が表示されています。これによって、必要に応じて緊急の電子メールのみを確認しやすくなります。このように、all ピボット項目の電子メールをスクロールして、緊急とマークされた電子メールを見つける必要はありません。



Outlook クライアント アプリ

アプリ タブ パターンを使う別の既存のアプリは、Netflix アプリです。各ピボット項目には、映画のリストが表示されます。各ピボット項目のリストは、適用されているフィルターによって異なります。比較することで、各リストがどのように独自のもので、どのように役立つかを確認できます。instant ピボット項目には、必要なときにストリーミング デバイスから視聴できるようにキューに配置した映画が表示されます。home ピボット項目には、ユーザーが興味を持つと Netflix によって判断された映画が表示されます。search ピボット項目には、検索キーワードに一致する映画が表示されます。

ピボット項目は、焦点が絞られており効率的です。主要な焦点は、関連する項目のフィルター処理、並べ替え、表示などのタスク ベースの操作です。ハブは範囲が広く、データが集められており、調査が必要です。最新の関連コンテンツを推進し、表示するのに最適です。特定のコントロールが他のコントロールより適している状況があります。

Netflix アプリ

タブのような機能

各ピボット項目は、実際にはタブであり、タブは上部に一覧表示されます。Netflix アプリの図からわかるように、アプリには現在 home ピボット項目が表示されており、右側には genres ピボット項目があることがわかります。genres ピボット項目に移動するには、タップするか、パンするだけです。

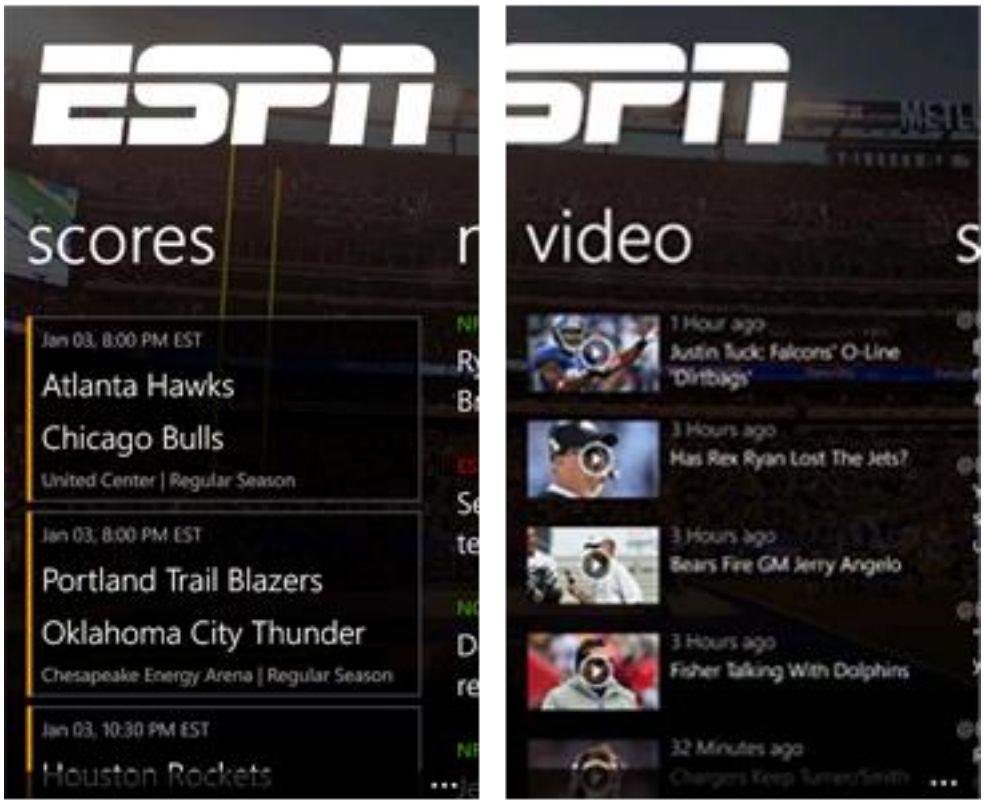
アプリでは、ピボット項目の数を最小限に抑えてください。ピボット項目からピボット項目にジャンプすると、ユーザーは迷う可能性があります。ピボット項目の数は、5 つ以下に抑えるようにしてください。それ以上必要なことがわかった場合、いずれかのピボット コントロールに、タップすると他のページやそれ自体がピボット コントロールである別のレベルのナビゲーションに移動するためのリンクが表示されたリスト コントロールを含めます。これが必要かどうかを考慮するには、「Windows Phone 用の詳細ドリルダウンを備えたリスト」をご覧ください。

注 ピボット コントロールを使うと、項目ヘッダーをタップして項目間を移動できます。これは、ハブ コントロールではできないことです。

データのフィルター処理についての考慮事項

このトピックの最初に、ピボット コントロールの最適な使用方法は、各ピボット項目に同じ種類のデータを含め、そのデータの異なるフィルター処理だけを表示することであると説明しました。さらに、関連しないデータを含むセクションを表示するには、ハブ コントロールを使うことができることも説明しました。データが本質的に異なっても、すべてのデータが何らかのテーマに関連していれば、ピボット コントロールをこのように使用することもできます。たとえば、ニューヨークに関する旅行者向けの情報をアプリに表示する場合、ピボット コントロールを使って、場所、人口、年間税収などの興味深い統計を含むピボット項目を表示することができます。その場合、見どころなどの情報のピボット項目と、お勧めホテルのピボット項目を追加することができます。

ESPN ScoreCenter アプリは、中心的なテーマに基づく異種データが各ピボット項目に含まれているアプリ タブ ナビゲーション パターンの例です。



ピボット コントロールとハブ コントロールを組み合わせる

ハブ コントロールの home ページ (Facebook アプリのホーム ページなど) からは、一覧でナビゲーション項目をタップして、ピボット コントロールに移動できます。これは、Facebook の home ハブ セクションから profile を選択したときに起きる事柄です。ユーザーが profile の選択をタップすると、wall、info、photos of me などのピボット項目を表示するピボット コントロールに移動します。このように、1 つのコントロールから他のコントロールに移動できます。

メイン ハブからピボットへ移動する

ハブの home セクションの選択から別のハブに移動したくなることがあるかもしれません。これは、ユーザーが現在使っているアプリを忘れやすくなって混乱する可能性があるため、お勧めしません。ユーザーが上部にいるときにユーザーを固定したままにするため、ハブの背景には大きく引き伸ばされた画像が表示されることがあります。それ自体がハブであり、背景が異なるサブ領域に移動した場合、ユーザーは混乱する可能性が高くなります。

ピボット コントロールのアプリ バー

すべてのデータが同じ種類であるため、多くの場合、表示されている内容に関連するボタンのある下部のアプリ バーを設置できます。次の図に示す Outlook アプリには、常に表示されるアプリ バーがあります。表示されている内容に何らかの方法で影響を与える、ユーザーに必要な操作が含まれているためです。

Outlook クライアント アプリ

Outlook アプリでは、ピボット項目にいるかどうかに関係なく、アプリ バーを使って同じ操作を行うことができます。前の図からは、新しいメールの作成、選択ボックスの表示、フォルダーの切り替え、メールの再同期を行うためのボタンがあることがわかります。この場合、これはピボット コントロールとハブ コントロールのもう 1 つの違いです。そこのボタンと同時にアプリ バーを配置することはおそらくないためです。

各ピボット項目に何を配置し、それに意味があるかを慎重し考慮しなければ、いつでも下部に同じボタンを表示し続けることはできないと考えてください。ピボット コントロールの使用方法によっては、各ピボット項目にまったく同じ種類のデータが維持されるわけではありません。

“Home” ピボット項目

最初のピボット項目を使って、残りのすべてのピボット項目の一覧を表示できます。こうすることで、ユーザーはスワイプする代わりにピボット項目をタップしてそのピボット項目に直接ジャンプできます。一覧には、他のアプリを起動するエントリが含まれていることもあります。

ユーザーを home ピボット項目に配置することを望まない場合は、そうする必要はありません。たとえば、映画リスト アプリでは、ユーザーを home ピボット項目に配置するのではなく、最新の人気映画のグラフィックが表示されるピボット項目にユーザーを配置できます。このようにして、より目を引く内容をユーザーに表示できます。

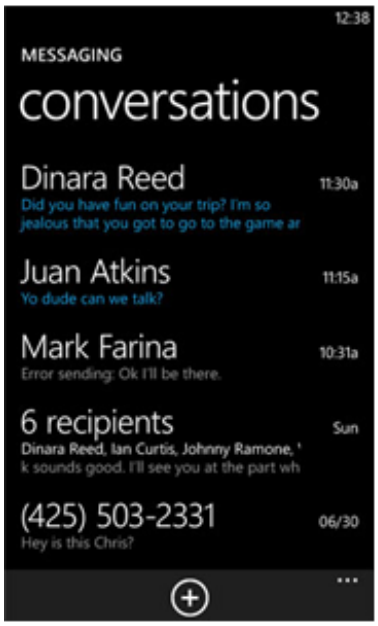
詳細ドリルダウンを備えたリスト

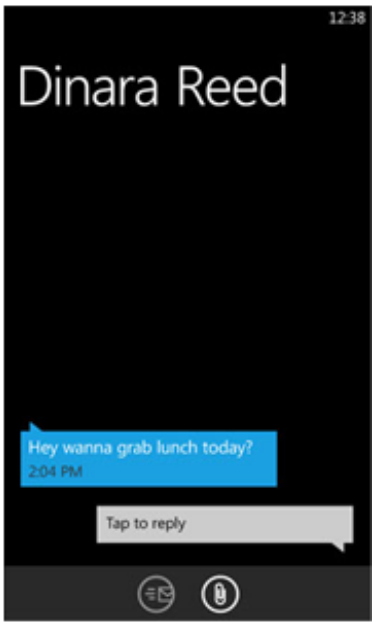
アプリに項目のリストを表示してユーザーがタップした際に詳細情報を表示する場合には、このパターンを使います。ページには項目のマスター リストが表示され、ユーザーはそこから項目を選んで詳細を表示します。このマスター リストには、各項目について必要なだけの情報を表示できます。単純なテキスト リストから画像のリストまで、さまざまなリストがあります。また、テキストと画像を組み合わせることもできます。

この単純なナビゲーション パターンでは、ピボット コントロールやハブ コントロールは必要ありません。アプリでは起動時に、リストのページが表示されるだけです。たとえば、ノーベル平和賞の受賞者の一覧を表示するアプリはこのパターンに従っています。リストの各エントリには、受賞者の写真、名前、国/地域などを含めることができます。任意の受賞者の情報をタップすると、その受賞者の詳細ページが表示されます。

注 ユーザーが選択できる項目が含まれる詳細ドリルダウンを備えたリスト パターンをアプリで使うと、多数の情報をスクロールして表示できます。

Windows Phone に付属のメッセージング アプリでは、このナビゲーション パターンが使われています。アプリを開くと、自分が参加した会話のリストが表示されます。他のページにはパンしません。このページで会話をタップすると、やり取りされた各メッセージが表示されるページに移動します。



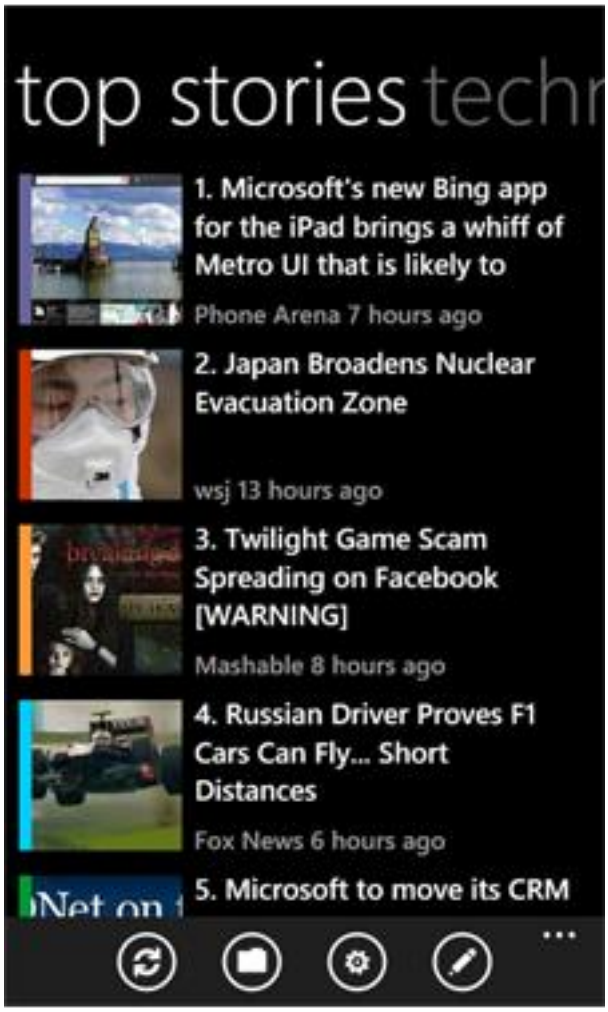


リストに画像とテキストが含まれる、この種類のモデルを使ったアプリを次の図に示します。

World Facebook アプリ

混合モデル

このナビゲーション パターンを他のメイン ナビゲーション パターンの 1 つに配置する場合があります。このリストとドリルダウン機能を備えたページは、ピボット コントロールやハブ コントロールのページの 1 つとして存在できます。これには、Project Emporia アプリが該当します。このアプリを開くと、既定で、[top stories (トップ ニュース)] と [world events (世界の事件)] の 2 つの項目がピボット コントロールで使用できます。これらの各ピボット項目ではナビゲーションのパターンが使われ、ユーザーは示される項目のリストから項目を選び、ドリルダウンすることでその詳細を表示できます。

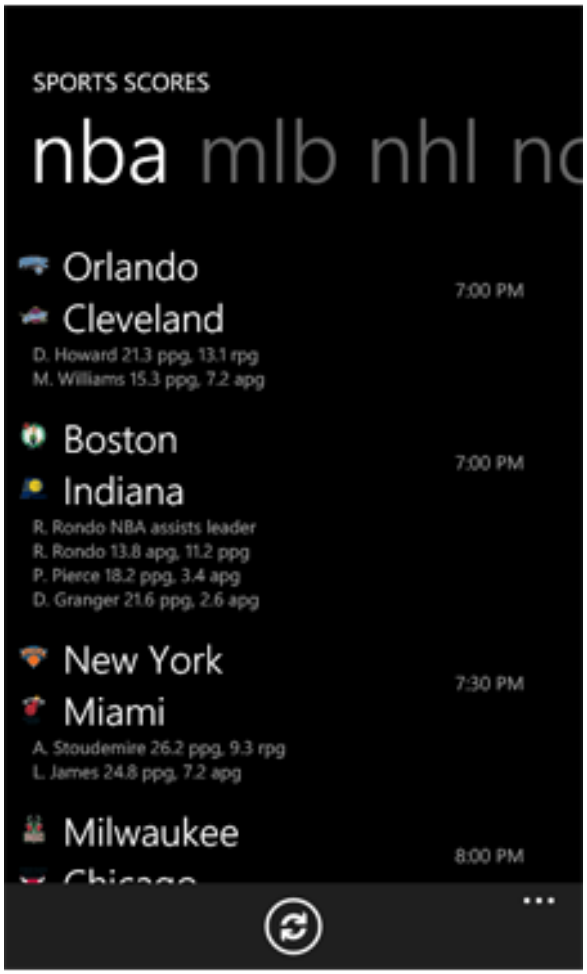


Project Emporia アプリ

均一化されたページ シャッフル

均一化されたページ シャッフル ナビゲーション パターンは、同じ形式と構造を持つ複数のページで異なる情報を表示する場合に適しています。フラッシュ カードのデッキのように、階層はなく、ページを単純にフリックでめくることができます。これを実装するには、pivot コントロールを使います。たとえば、お気に入りのチームのベースボール カードの画像を表示するアプリを作るとします。各ページでは、同じレイアウトでプレーヤーの写真、名前、統計情報を表示します。

このようなアプリは、使いやすく、すべてのページ領域をデータを表示するためのコンテンツに使うことができます。次の例は、ページ間を移動できるアプリで、異なるデータについて同じ情報レイアウトを表示しています。この例では、複数のスポーツ リーグのスコアを表示しています。



スポーツ スコア アプリ

常にすべてのページを表示する必要があるユーザーのために、並べ替えできるようにすることも可能です。たとえば、天気予報アプリで、フリップで表示できる都市の順序をユーザーが並べ替えられるようにします (住んでいる都市の天気を表示した後、好きな観光地の天気を表示するなど)。

同じ種類のデータのインスタンスが多数あり、階層や複雑なデータ ナビゲーションが必要ない場合は、このナビゲーション パターンを使います。ページを追加して設定することが簡単にできます。

ユーザーの構成

均一化されたページ シャッフル ナビゲーション パターンを使った便利なアプリでは、ほとんどの場合、ユーザーが表示するページを追加できるようにする必要があります。天気予報アプリでは、ユーザーが表示する都市を追加する場合が多くあります。これを考慮し、そのための設定ページを用意する必要があります。また、ユーザーが不要になったページを削除できる機能を追加することも忘れないでください。このような操作を行うために、下部のアプリ バーを使うことができます。

コマンド パターン

アプリ キャンバス、フライアウト、ダイアログ、アプリ バーなど、Windows ストア アプリの数種類のサーフェスにコマンドとコントロールを配置することができます。適切なタイミングで正しいサーフェスを選ぶことは、簡単に使用できるアプリとなるか、負担の多いアプリとなるかの分かれ目になります。

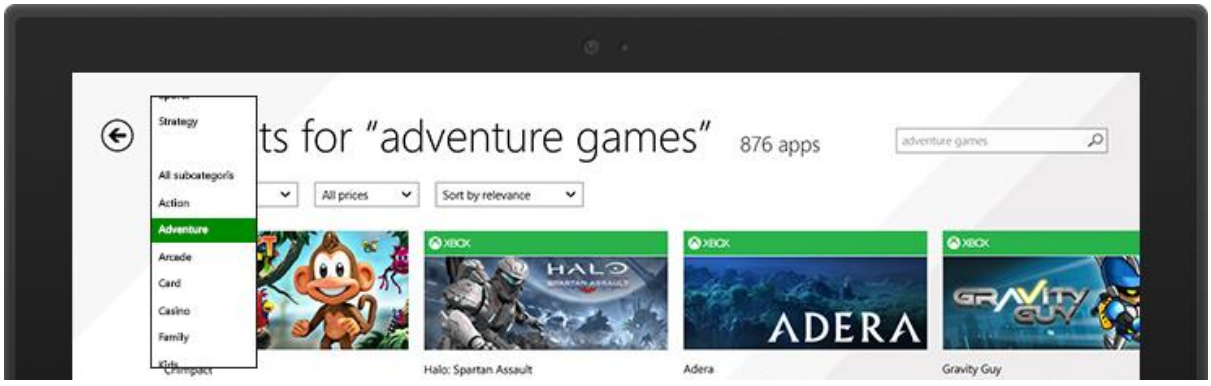
Windows ストア アプリでは、コマンドはユーザーが操作するために使うことができる対話型の UI 要素です。コマンドは、ナビゲーション要素とは異なります。ナビゲーション要素ではユーザーは別のページに移動しますが、コマンドでは現在のページで操作を実行できます。ナビゲーション要素によってアプリを使うことができるようになり、コマンドによってアプリが便利になります。

Windows ストア アプリのさまざまなコマンド サーフェスについて詳しくは、「UI のレイアウト」をご覧ください。

コマンドの種類

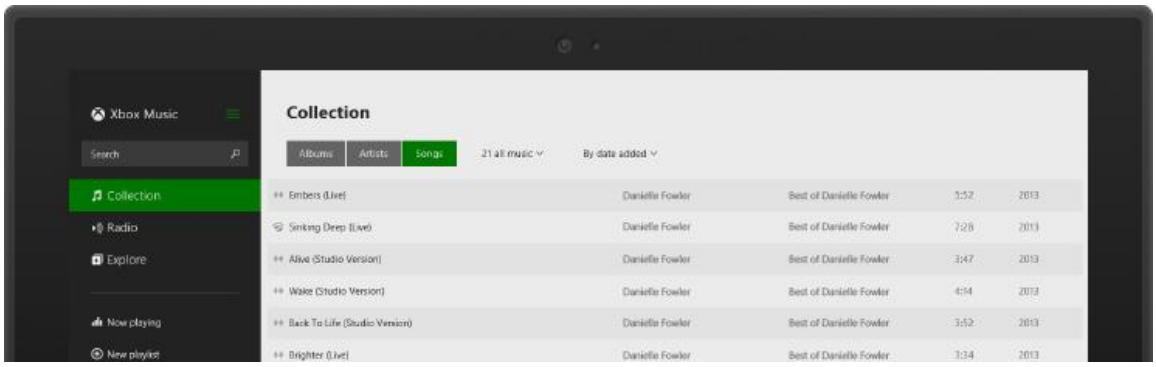
フィルター

フィルター コマンドは、一定の基準に基づいて、データ セット内のコンテンツを削除または非表示にします。たとえば、ユーザーは、Windows ストアで、"アドベンチャー" に分類されるゲームのみを表示できます。



ピボット

ピボット コマンドは、ピボットに基づいて、データ セット内のコンテンツを再編成し、データの異なるビューを表示します。たとえば、ユーザーは楽曲をアルバム別、アーティスト別、曲名別に表示できます。

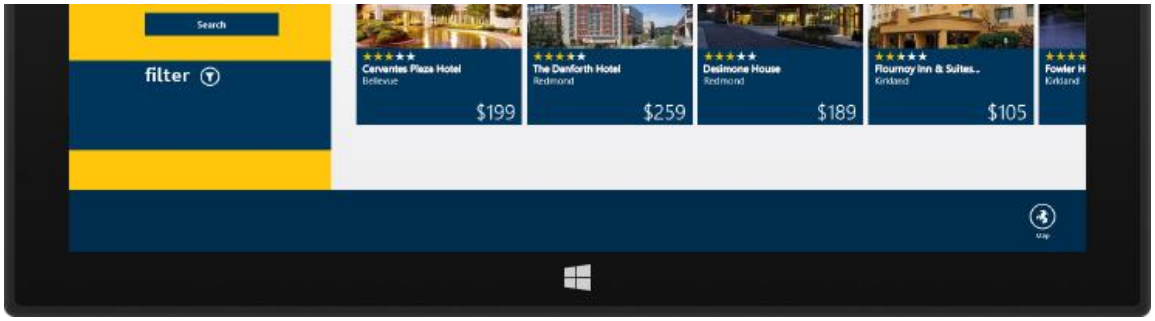


並び替え

並べ替えコマンドは、データ セット内のコンテンツの表示順序を変更します。たとえば、トラベル アプリでユーザーは旅行先を人気順で表示できます。

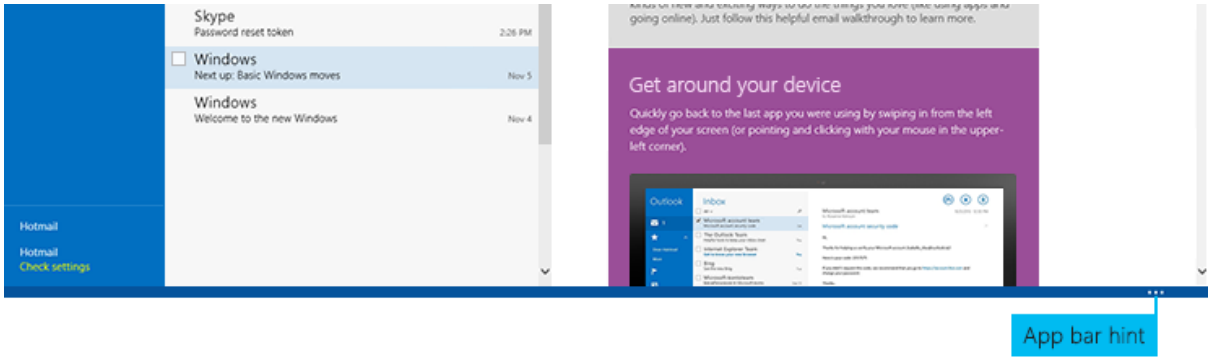
表示

表示コマンドは、コンテンツを表示するスタイルや方法を変更します。たとえば、ホテルを検索するアプリで、ユーザーは一覧ではなく、地図上にホテルを表示できます。

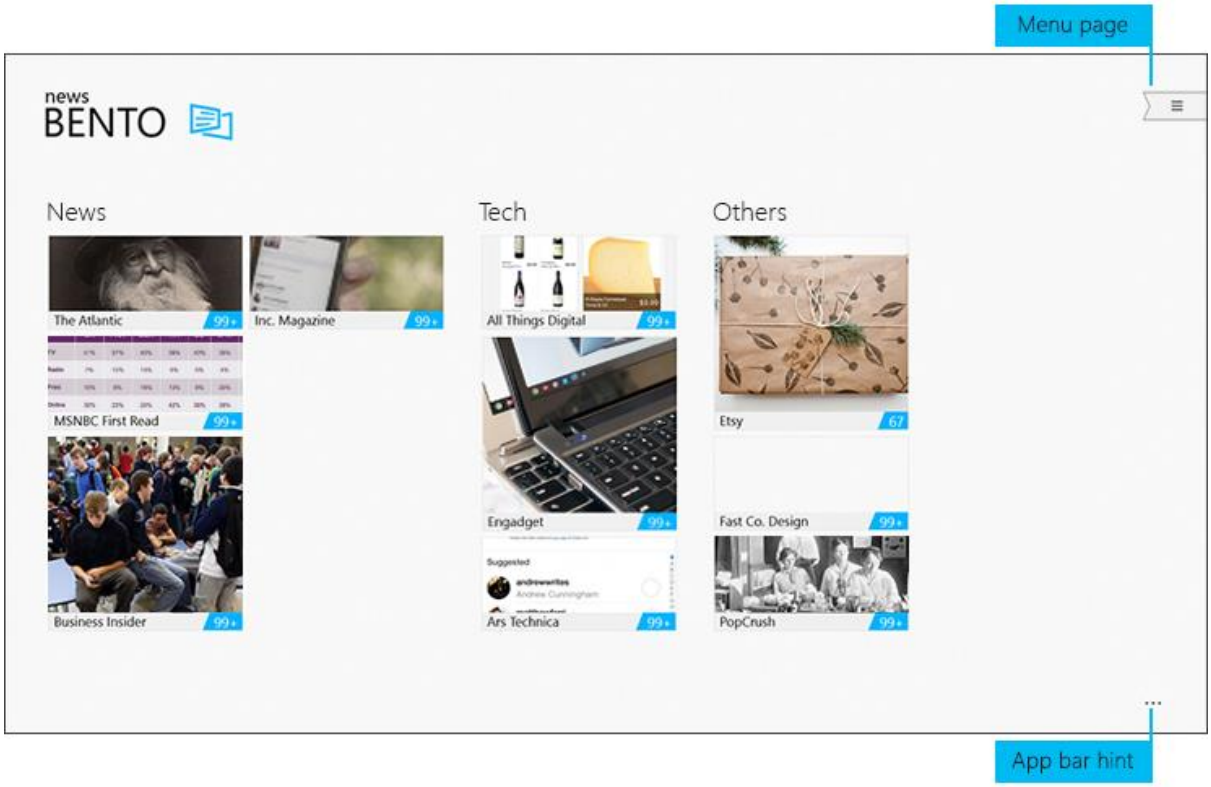


ヒント

ユーザーに役立つようであれば、キャンバスにヒント コマンドを配置できます。たとえば、Windows 8.1 で動作するメール アプリには、追加のオプションを示す"省略記号"のコマンドがあります。Windows でアプリ バーを表示していた従来の方法に加えて、このコマンドをクリックしても、アプリ バーが表示されます。

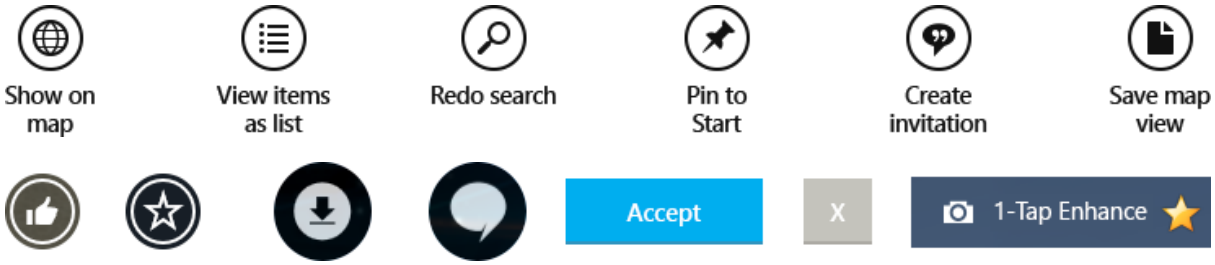


アプリのスタイルに合ったヒントを設計できます。このニュース アプリでは、同じアプリバー ヒントに加えて、メニュー選択肢のページを表示する旗のメニュー アイコンも使っています。



その他

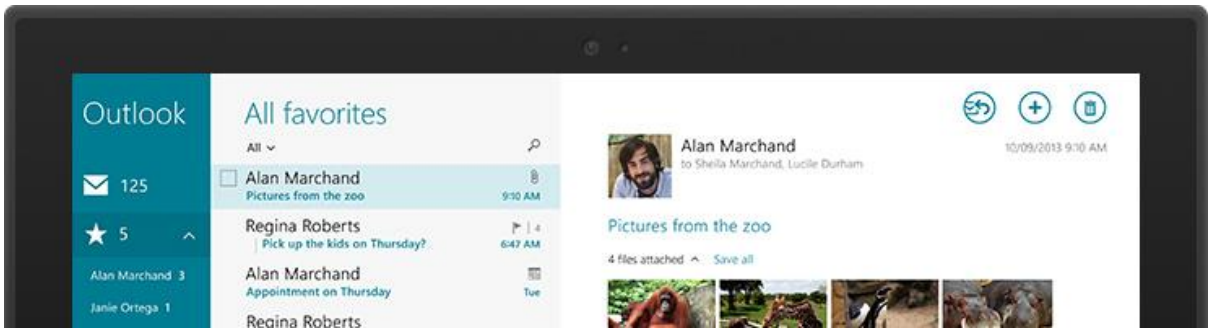
現在のビューでユーザーが操作するために使うことができる対話型の UI 要素は、すべてコマンドです。次にいくつかの例を示します。



コマンドの配置

キャンバス上

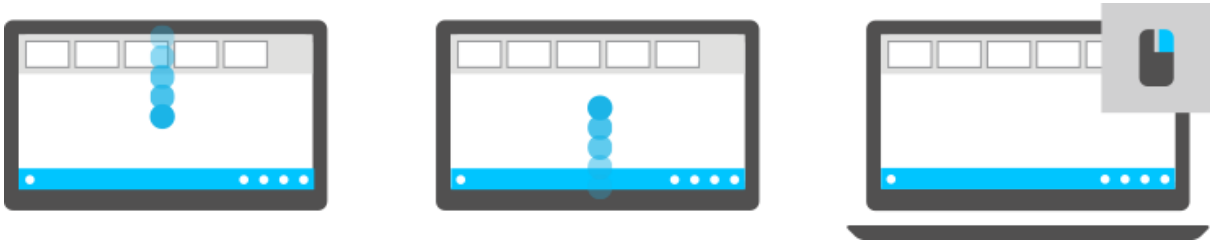
あるコマンド (またはコマンド メニュー) が重要で、ユーザーが中心的なシナリオを完了するうえで常に必要な場合は、そのコマンドをキャンバスに配置できます。たとえば、Windows 8.1 のメール アプリでは、メールを選ぶと、返信、新規作成、削除などの中心的なコマンドがキャンバスに表示されます。



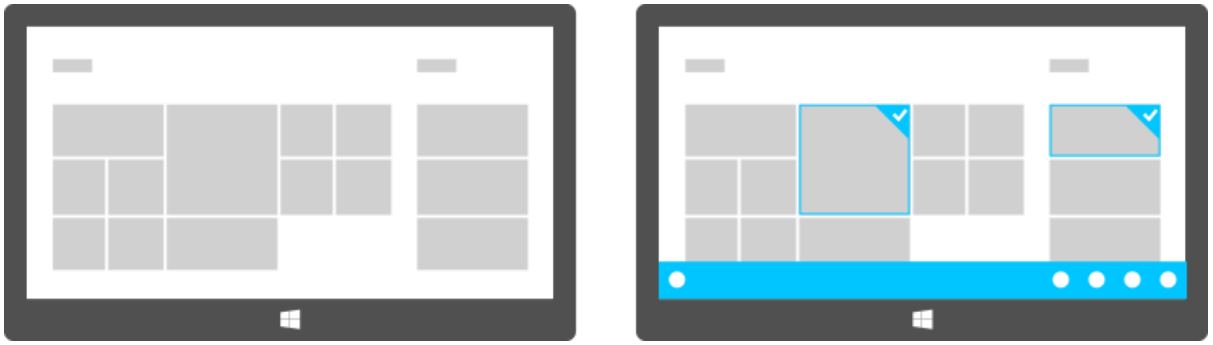
下側のアプリバー

キャンバスがコマンドで乱雑になることや、コマンドによってアプリのコンテンツを利用しづらくなることを回避する必要がある場合があります。下側のアプリ バーを使うと、コマンドをオンデマンドでユーザーに表示することができます。下側のアプリ バーには、ユーザーのコンテキスト (通常は現在のページや現在の選択) に関連するコマンドが表示されま

す。下部のアプリ バーは、既定では表示されません。このアプリ バーは、ユーザーが指で画面の上端または下端をスワイプしたとき、または右クリックしたときに表示されます。



下側のアプリ バーは、単一選択のコマンド実行でも複数選択のコマンド実行でも正しく動作します。ユーザーがオブジェクトを選ぶとアプリ バーが表示されるように、アプリのプログラミングを行うことができます。



コマンド配置のガイドライン

さまざまな方法でアプリ バー内にコマンドを配置できますが、次の点を考慮する必要があります。

- **予測可能性** できる限り、アプリのすべてのビューを通じて、一貫した対話的操作とコマンド配置を使います。
- **人間工学** コマンドの配置によって操作がどのようにすばやく簡単にできるようになるかを考慮します。
- **デザイン性** アプリ バーが複雑にならないように、コマンドの数を制限します。すぐに理解でき、機能を推測できるようなアイコンを選びます。テキスト ラベルを短くします。

次のコマンド配置のガイドラインに従うことをお勧めします。

永続的なコマンドまたは既定のコマンドをアプリ バーの右側に配置します。コマンドの数が少ない場合、アプリ バーは右側だけにコマンドがある状態になります。
この例では、参照コマンド用に、表示コマンド セットとフィルター/並べ替えセットが永続的になっています。

エッジ（端）を使います。コマンドの数が多い場合、それぞれのコマンド セットを左側と右側に分けて、アプリ バーのバランスを取り、人間工学的にコマンドを利用しやすくします。
この例では、表示コマンド セットを左側に移動し、フィルター/並べ替えセットは右側に残すことにします。 地図ビューがアクティブな場合、地図ビュー コマンドが表示コマンド セットの右側に表示されます。

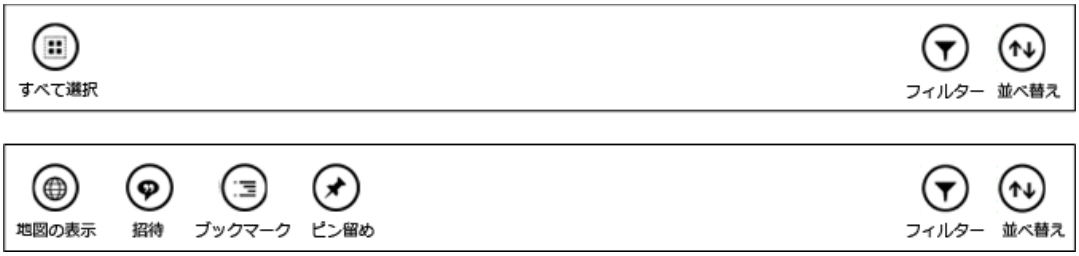
無効なコマンドを表示または非表示にします。特定の状況に無関係なコマンドがあります。それらのコマンドを表示する場合は、永続的なコマンドの順序を邪魔しないようにします。
この例では、地図ビューがアクティブで、地図ビュー コマンドが表示コマンド セットの右側に表示されます。

選択コマンドを挿入します。ユーザーが選択を行った結果として表示されるコマンドは、左端に移動され、そこにあったコマンドはスライドします。これによって、選択コマンドがより目立ち、アクセスしやすくなります。
この例では、表示コマンド セットが右側にスライドして、選択コマンド セットのための場所を空けます。

一般的なコマンドの配置

いくつかのコマンドは、多くのアプリで共通して表示されます。一般的なコマンドをアプリバーのどこに配置するかを決めるときには、一貫性を保ち、確実性を高めるために、次のガイドラインに従うことをお勧めします。

選択コマンドは左端に配置します。 それが選択時に表示される状況依存のコマンドであるか、 選択方法に影響するコマンドであるかは関係ありません。
この例では、ユーザーが何かを選択する前に、[すべて選択] コマンドが左側に表示されます。ユーザーが何かを選択すると、他の選択コマンドが左側に表示されます。



新規項目コマンドをバーの右端に配置します。新規項目コマンドは、追加、作成、構成のコマンドである場合や、新しいエンティティを作る任意のコマンドである場合もあります。新規項目コマンドは、親指で簡単にアクセスできる必要があります。
この例では、[新しいレビュー] コマンドで、ユーザーはレストランの新しいレビューを作成できます。 [新しいレビュー] に関連する他のコマンドは、左隣に配置されます。
+ 記号は "新規" コマンドを表す場合だけに使い、アプリ バーの他の場所には表示されないようにします。

削除コマンドを新規項目コマンドの左側に配置します。メール アプリやカメラ アプリのように、特定のアプリケーションの外部で永続する個々のエンティティをアプリが管理する場合は、削除/新規を使います。 削除/新規は、常にこの順序で表示します。

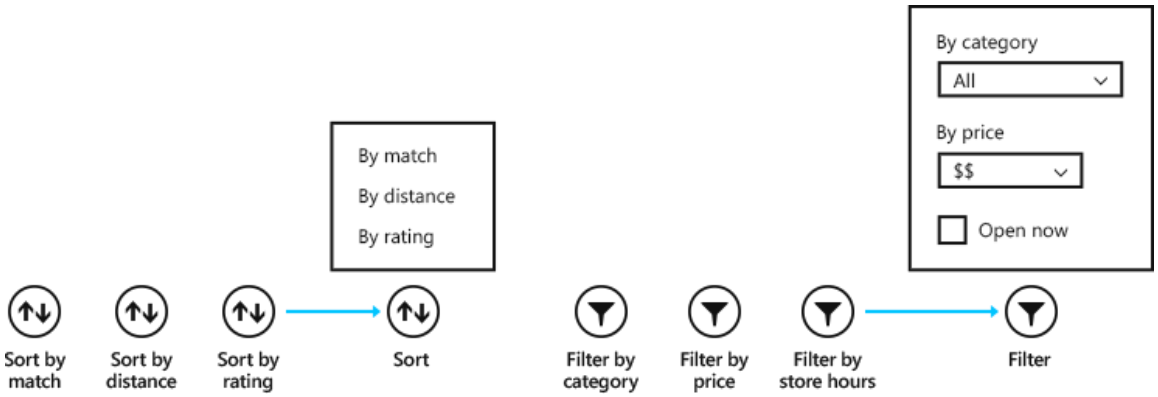
除去コマンドを追加コマンドの左側に配置します。To Do リスト、天気予報アプリの都市のリスト、ブックマークが付けられたレストランのリストなど、アプリがリストを管理する場合は、除去/追加を使います。除去/追加は、常にこの順序で表示します。

消去コマンドを新規項目コマンドの左側に配置します。すべての可能な項目に対して破壊的な操作を行う場合は、消去を使います。"選択項目の消去" のように、何がコマンドの操作対象になるかを明確に示すコマンド ラベルを使います。

コマンドをメニューにグループ化する

複数のコマンドをコマンド メニューにグループ化することで効率性が高まる場合があります。 メニューを使うと、より狭い場所により多くのオプションを表示できます。メニューには対話的なコントロールを含めることができます。

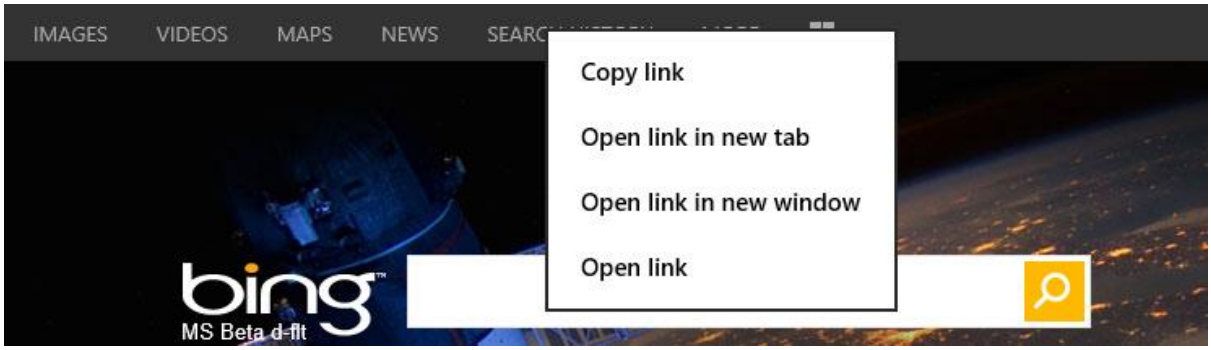
この例では、並べ替えコマンドに、ユーザーが並べ替えオプションを簡単に選べるシンプルなメニューを表示します。フィルター コマンドでは、より複雑な条件で項目をフィルター処理できる一連のコントロールをメニューに表示します。



コンテキスト メニュー

多くの場合、コンテキスト メニューにはクリップボードの操作 (切り取り、コピー、貼り付けなど) が含まれます。また、コンテキスト メニューには、選ぶことができないコンテンツ (Web ページ上の画像など) に適用するコマンドを含めることもできます。 アプリでは、システムに用意されているテキストとハイパーリンクに関する既定のコンテキスト メニューを使うことができます。テキストの場合、既定のコンテキスト メニューにはクリップボード コマンドが表示されます。ハイパーリンクの場合、既定のコンテキスト メニューには、リンクをコピーするコマンドとリンクを開くコマンドが表示されます。

ユーザーは、タッチ デバイスでコンテンツを長押しするか、マウスでコンテンツを右クリックして、コンテキスト メニューを呼び出します。



チャームの活用

Windows ストア アプリを設計する場合は、検索、共有、デバイス、設定というチャームの 4 つの非常に便利なコマンドを使います。ユーザーは、画面の右端からスワイプするか、画面の上端または下端の右隅に向かってマウス カーソルをポイントして、チャームを起動します。

検索: ユーザーはアプリのコンテンツを、システムのあらゆる場所から (別のアプリからも) すばやく検索できます。

共有: ユーザーはアプリのコンテンツを他のユーザーやアプリと共有できるだけでなく、共有コンテンツを受け取ることもできます。

スタート: スタート チャームは、スタート画面を表示します。アプリがこのチャームを操作することはありません。動作は自動的で、常に同じです。

デバイス: ユーザーはオーディオ、ビデオ、または画像をアプリからホーム ネットワーク上の別のデバイスにストリーミングして 楽しむことができます。

設定: すべての設定を 1 つの場所にまとめて、ユーザーが使い慣れた一般的な方法でアプリを 構成できます。

アプリのキャンバスまたはアプリ バーでアプリ コントラクト機能が重複しないようにします。

入力とフィードバックのパターン

タッチ操作に対応したアプリを設計する場合、タッチパッド、マウス、ペン、キーボードの操作が標準でサポートされます。ユーザーが入力方法を切り替えても、アプリの使いやすさは変わりません。タブレットにキーボードを接続しても、問題ありません。アプリは、ユーザーの選択に対して一貫して予想どおりに応答します。

タッチに対応した Windows ストア アプリの UI を設計します。ただし、さまざまなデバイスに対する設計上の影響を検討してください。

- タッチとマウスのエクスペリエンスを組み合わせるタッチパッド
- マウス
- デジタル インク操作に特化したペン
- キーボード デバイス

タッチ

Windows 8 には、システム全体で使われるタッチ操作の簡単なセットが用意されています。このタッチ言語を一貫して適用することで、ユーザーに、自分が使い慣れたアプリと似ていると感じさせることができます。アプリの使い方をわかりやすくすることで、ユーザーの信頼感が増します。

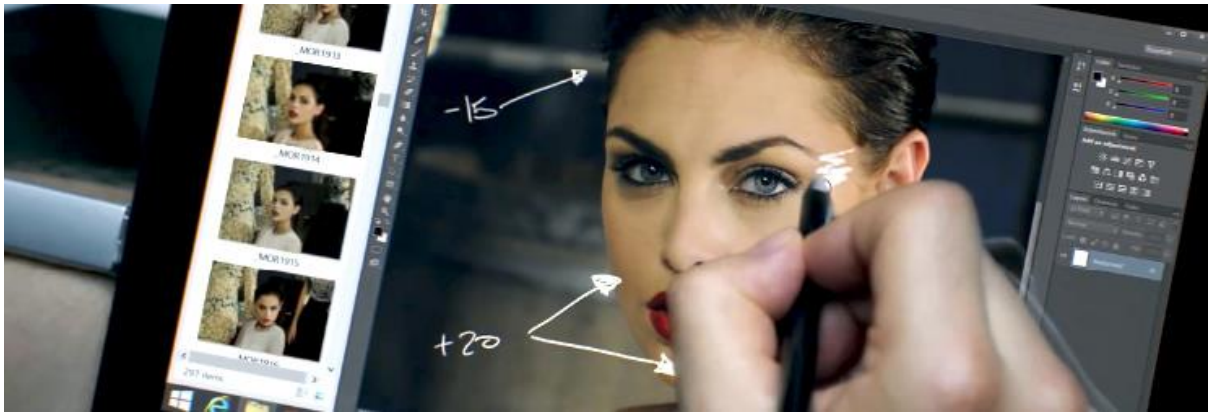
タッチパッド

タッチパッドは、間接的なマルチタッチ入力と、マウスのようなポインティング デバイスの精密入力を組み合わせたものです。この組み合わせにより、タッチパッドはタッチに最適化された UI にも、生産性アプリとデスクトップ環境のより小さいターゲットにも適しています。Windows ストア アプリの設計はタッチ入力用に最適化し、標準として提供されるタッチパッドのサポートを利用します。

マウス

マウス入力は、ポイントとクリックの正確さが求められるユーザー操作に最適です。この固有の正確さは、Windows 8 の UI でも当然サポートされていますが、タッチの本来の不正確さに合わせて最適化されています。Windows ストア アプリの設計はタッチ入力用に最適化し、標準として提供されるマウスのサポートを利用します。

ペン



ペンは、ポインティング デバイスとして使うことができます。また、デジタル インクに関連付けられた描画デバイスにすることもできます。

Windows 8.1 のインク プラットフォームでペン デバイスを使うと、自然な形で手書きノート、描画、コメントを作成できます。このプラットフォームは、デジタイザー入力からのインク データのキャプチャ、インク データの生成、出力デバイスへのインク ストロークとしてのデータのレンダリング、インク データの管理、手書き認識の実行をサポートします。

キーボード

キーボード操作により、アプリが使いやすくなる可能性があります。アクセシビリティのために重要であり、タッチ スクリーンがない場合に機能を使うためにも重要です。さらに、キーボードをサポートすると、キーボードをよく使うユーザーがアプリを使いやすくなります。ユーザーが、Tab キーと方向キーを使ってアプリ内を移動し、Space キーと Enter キーを使って UI 要素をアクティブ化し、キーボード ショートカットを使ってコマンドにアクセスできる必要があります。

ユーザー エクスペリエンスの強化

ユーザー操作に関する UX ガイドラインに従うときは、創造性を大切にしてください。アプリでサポートする入力デバイスと、アプリで入力に反応する方法を選びます。デバイス間のユーザー エクスペリエンスを強化すること、できるだけ幅広い機能と設定をサポートすること、Windows ストアでできるだけ多くの潜在顧客にアピールすることを心がけてください。そうすれば、アプリの使い勝手、移植性、アクセシビリティが最大限に高まります。

Windows のタッチ操作

Windows 8.1 には、システム全体で使われるタッチ操作の簡単なセットが用意されています。このタッチ言語を一貫して適用することで、ユーザーに、自分が使い慣れたアプリと似ていると感じさせることができます。アプリの使い方をわかりやすくすることで、ユーザーの信頼感が増します。

タッチの設計原則

即座にフィードバックを返す

画面がタッチされたときにすばやくビジュアルなフィードバックを返すようにすると、ユーザーが安心して操作できます。操作に対して、色やサイズの変更、移動で応答する必要があります。操作できない要素は、画面がタッチされたときだけタッチ画像を表示するようにします。

コンテンツが指の動きに付いていくようにする

キャンバスやスライダーなど、ユーザーが移動またはドラッグできる要素は、画面上の指の動きに付いていく必要があります。移動できないボタンなどの要素は、指を離したときに既定の状態に戻る必要があります。

逆方向の操作ができるようにする

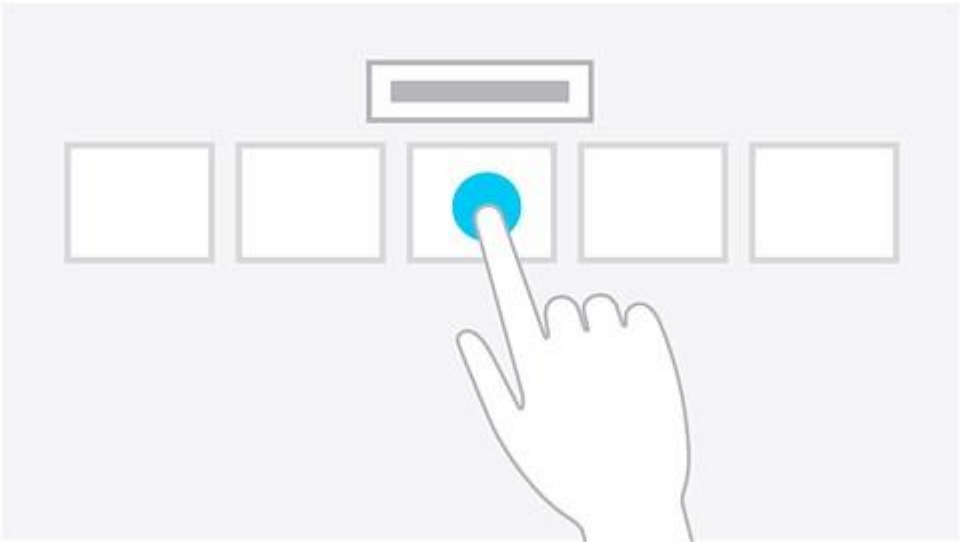
本棚から取り出した本は、元の場所に戻すことができます。それと同じように、タッチ操作は元に戻すことができることが必要です。ユーザーが指を離したときに何が起こるかを、ビジュアルなフィードバックで示します。これによって、タッチ操作で安心してアプリを使用できるようになります。

複数の指を使うことができるようにする

ユーザーは意識せずに複数の指を使ってタッチしていることがよくあります。そのため、画面をタッチする指の数が変わったときにタッチ操作を大きく変化させないようにしてください。 現実世界と同じように、1 本の指でスライドするときも、3 本の指でスライドするときも、結果が変わらないようにしてください。

タッチ言語

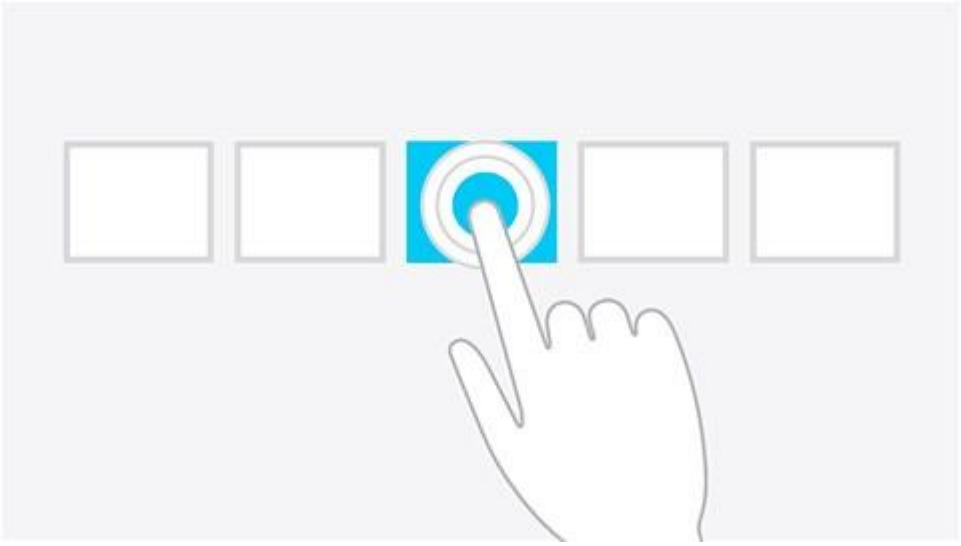
長押しによる情報の表示



このタッチ操作では、詳細情報や説明を伝える視覚効果 (ツールチップやコンテキスト メニューなど) が表示されます。操作を確約する必要はありません。ユーザーが指でスライドを開始したときに、このような表示によってパンが妨げられないようにしてください。

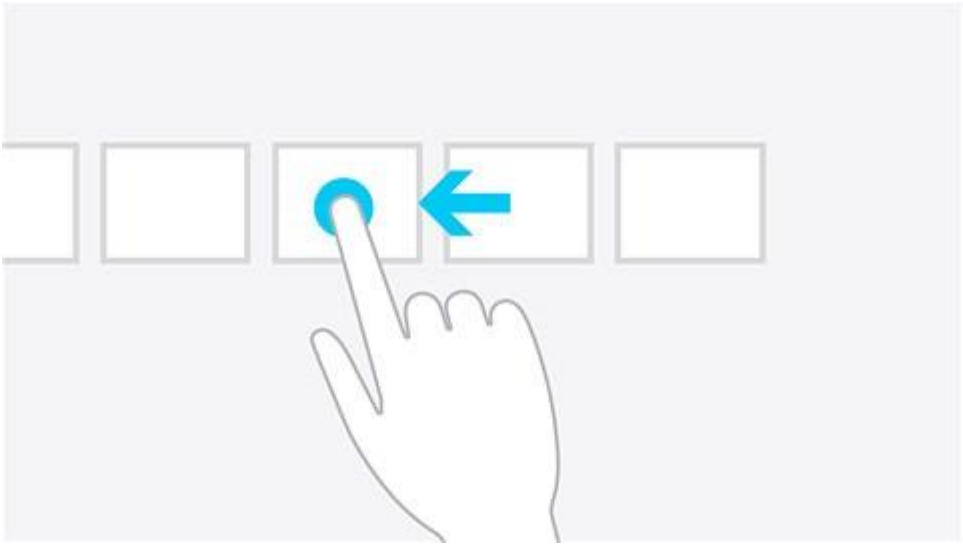
重要 水平方向と垂直方向のパンが有効である場合は、長押しを選択に使うことができます。

タップによるプライマリ 操作



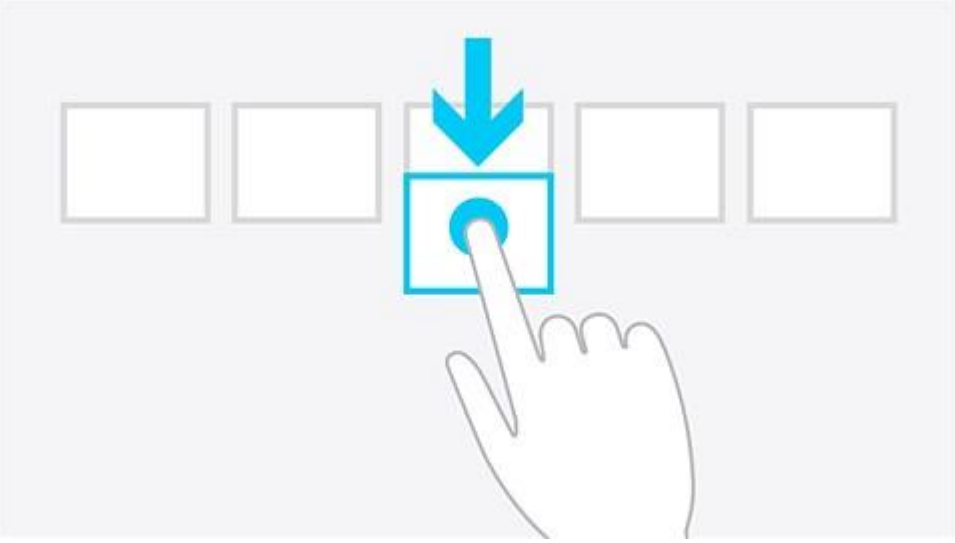
要素をタップするとプライマリ操作 (アプリの起動、コマンドの実行など) が呼び出されます。

スライドによる パン



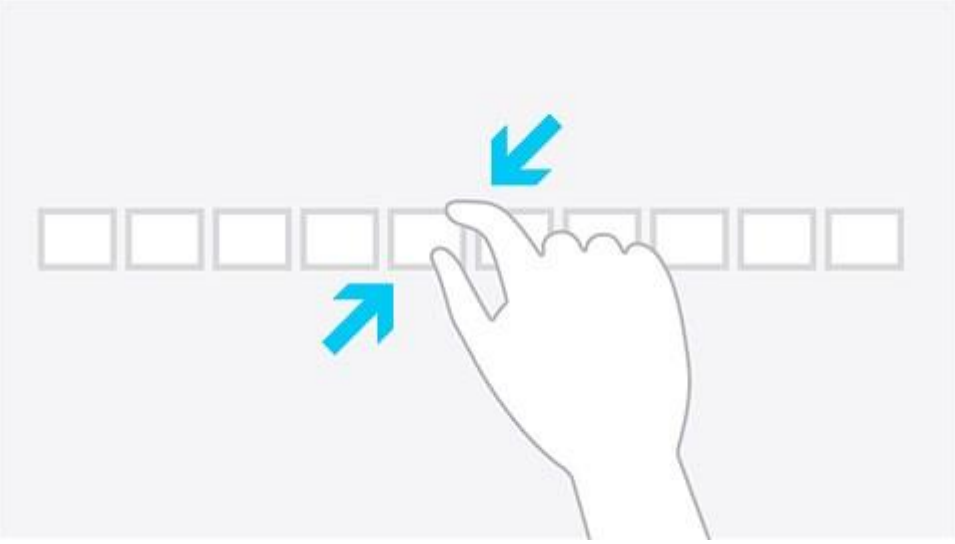
スライドは主にパン操作に使われますが、移動 (パンの方向が 1 方向に制限されている場合)、描画、筆記などの操作に使うこともできます。 また、スライドは、スクラブ操作 (ラジオ ボタンなどの関連したオブジェクトを指でスライドする操作) によって、小さな密集した複数の要素をターゲットとするときにも使うことができます。

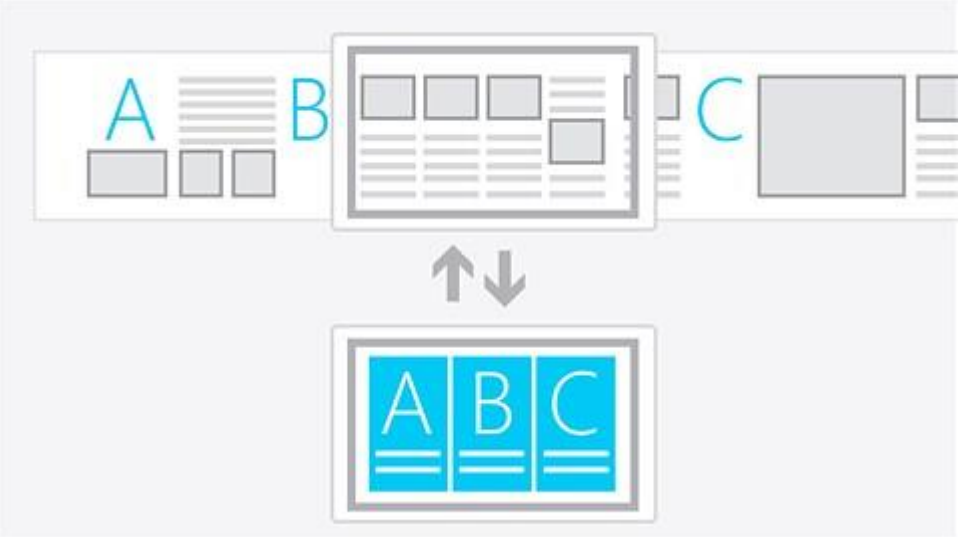
スワイプによる選択、コマンド、 移動



パン方向に対して 垂直方向に指を少しスライドさせると (パンの方向が 1 方向に制限されている場合)、一覧またはグリッドの中の オブジェクトが選択されます。オブジェクトが選択されると、関連するコマンドを含むアプリ バーが 表示されます。

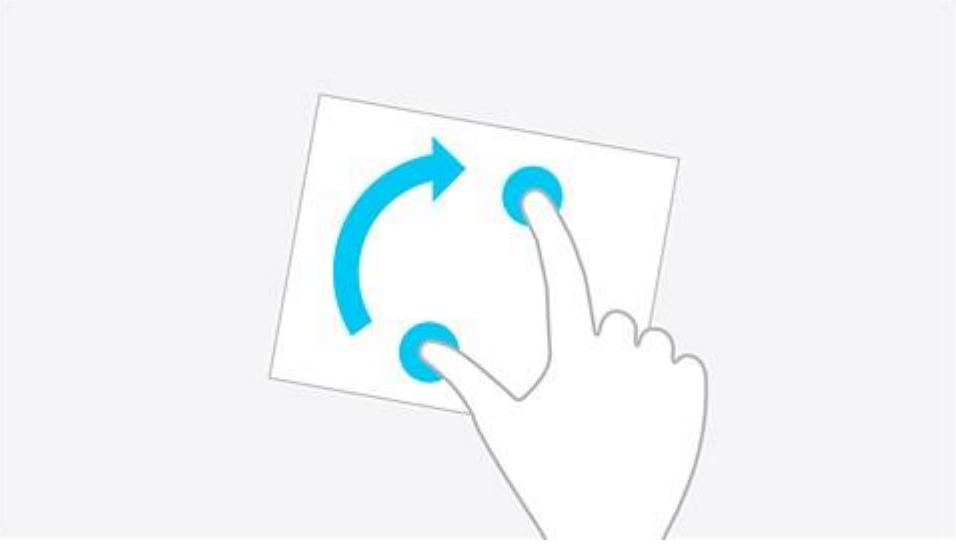
ピンチとストレッチによる ズーム





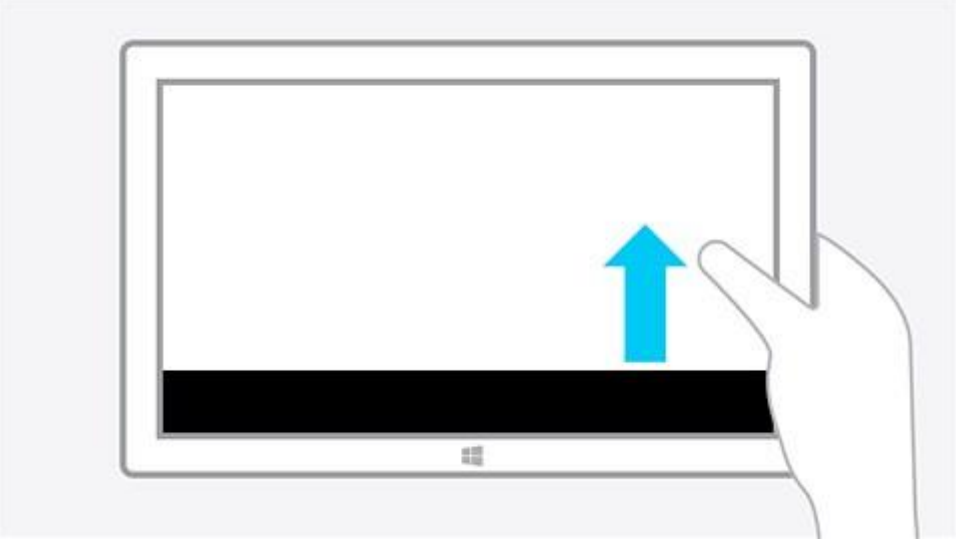
ピンチ ジェスチャーとストレッチ ジェスチャーは 一般的にサイズ変更に使われますが、セマンティック ズームによってコンテンツ内の最初、最後、または 任意の場所に移動することもできます。セマンティック ズームは、アイテムのグループを表示し、 それにすばやく戻ることができるズーム アウト ビューを 提供します。

指を違う方向に動かすことによる回転



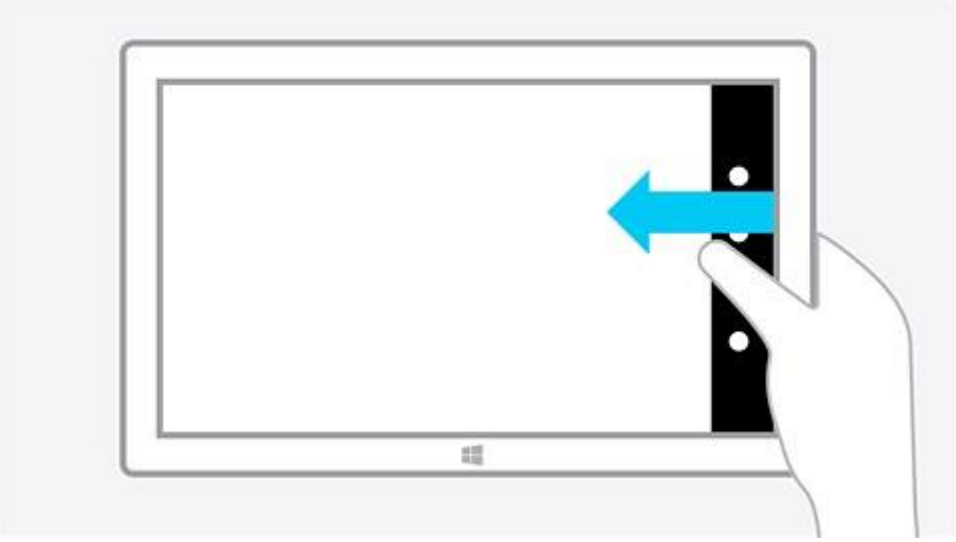
2 本以上の指を別々の方向になぞると、オブジェクトが回転します。画面全体を回転させるには、デバイスを回転させます。

エッジからのスワイプによるアプリ コマンドの 表示



画面の上端または下端からスワイプすると、 アプリ コマンドが表示されます。アプリ バーを 使ってアプリ コマンドを 表示します。

エッジからのスワイプによるシステム コマンドの 表示



画面の右端から スワイプすると、システム コマンドが含まれた チャームが表示されます。

左端からスワイプすると、現在実行中のアプリが順に表示されます。

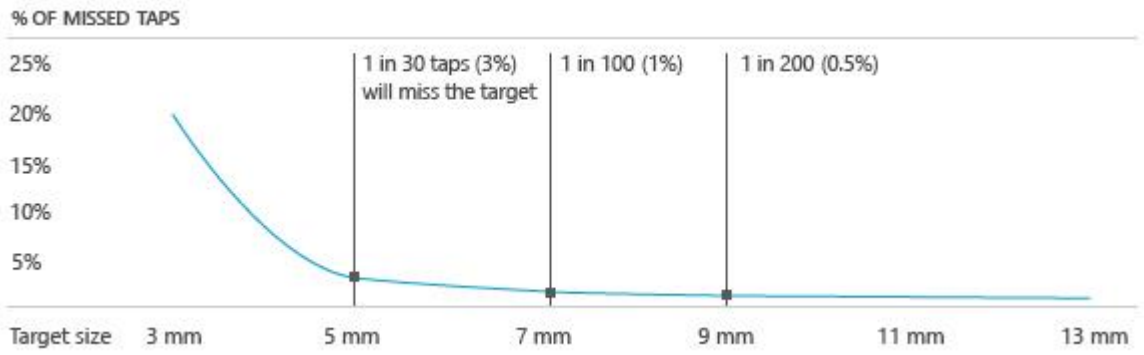
画面の上端から下端へスライドすると、現在のアプリを閉じます。

画面の上端から少し下げて、左端または右端へスライドすると、 現在のアプリをスライドした側に配置します。

注 ユーザーは、スライドによるパン、ピンチによるズーム、指を違う方向に動かすことでの回転などの直接的な操作を、多数のタッチ ポイントで同時に行えます。

タッチ ターゲット

サイズと効率: ターゲットのサイズは誤操作の発生率に影響する

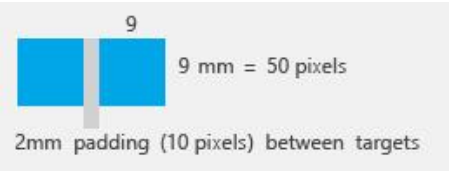


タッチ ターゲットには、完ぺきなサイズというものはありません。状況が異なれば、有効なサイズも異なります。削除、閉じるなど、大きな変化が起こる操作や、よく使う操作には、大きなタッチ ターゲットを使う必要があります。大きな変化が起きない、あまり使われない操作には、小さなターゲットを使うことができます。

ターゲットのサイズの ガイドライン

7x7 mm: お勧めの最小サイズ

タッチ ミスを 1 回または 2 回のジェスチャーで修正できる場合、または 5 秒以内で 修正できる場合は、7 x 7 mm はお勧めの最小サイズです。ターゲット間の余白は、ターゲット サイズと同じくらい重要です。



正確さが重視される場合

閉じる、削除など、大きな変化が起こる操作には タッチ ミスが許されません。タッチ ミスの修正に 2 回以上のジェスチャーが必要な場合、 5 秒よりかかる場合、またはコンテキストの大幅な変更が 必要な場合は、9 x 9 mm のターゲットを使います。

入りきらない場合

どうしても範囲内に入りきらない場合は、 タッチ ミスを 1 回のジェスチャーで 修正できるなら、5x5 mm のターゲット を使ってもよいでしょう。この場合、 ターゲット間に 2 mm の余白を 入れることが、非常に重要になります。

アクセシビリティ

アプリでサポートされる対話操作と UI を計画する際は、ユーザーによってできる操作、できない操作、好ましい操作が大きく異なることを心に留めておいてください。初めからアクセシビリティに対応した設計原則に従っておくことで、アプリの対象ユーザーの間口が広がります。アクセシビリティの計画について詳しくは、「アクセシビリティのガイドライン」をご覧ください。

指のサイズ

"指が太い" と自分を責める人が よくいます。しかし、 赤ちゃんの指でさえ、タッチ ターゲットより大きいことがよくあります。

下の図によると、成人の平均的な指の太さは約 11 mm で、赤ちゃんは 8 mm です。バスケットボールの選手は 19 mm を超える場合もあります。

位置と持ち方

タッチの設計は、画面上のオブジェクトの設計よりも重要です。また、デバイスの握り方 (持ち方) も考慮する必要があります。

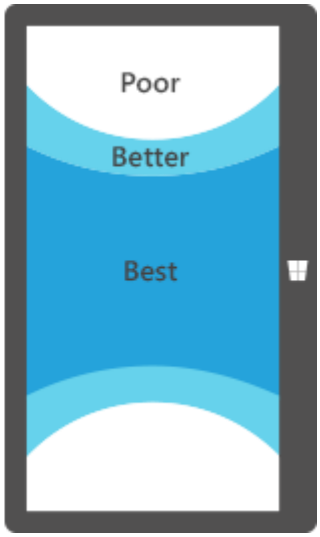
一般的に、ユーザーによって、タブレットを使う際のお気に入りの持ち方がいくつかあります。

現在の作業と、その表示方法によって、持ち方が変わってきます。それだけでなく、周囲の状況や使い勝手によっても、その持ち方がどのくらいの時間続くのか、どの程度の頻度で変わるのかが変わってきます。

さまざまな持ち方に対応できるようにアプリを最適化してください。ただし、ある操作をするときは自然に特定の持ち方になる場合は、その持ち方に対して最適化します。

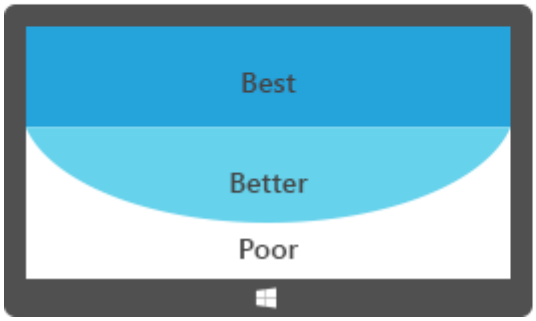
操作領域

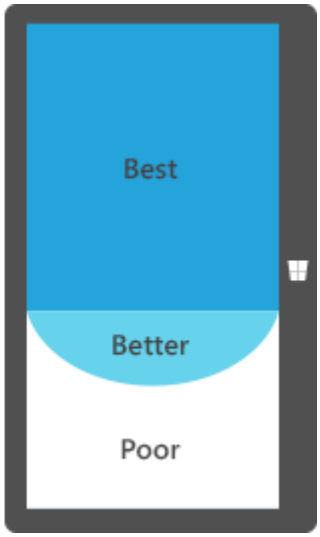
スレートは 両端を持つことが多いため、操作する要素は下の隅と、 左右の端に置くのがお勧めです。



読み取り 領域

画面の上半分のコンテンツは、手で覆われたり無視されたり しやすい下半分のコンテンツよりも読みやすくなります。





よく使われる 持ち方

持ち方	持ち方と操作	設計時の考慮事項
	片手で持ち、もう一方の手で簡単な操作から中程度の操作まで行う	•右端または下部に配置したオブジェクトは、すばやく操作できます。 •右下隅は手や手首でふさがれる場合があります。 •届く範囲が制限されるため、タッチが正確になります。 •読み取り、閲覧、メール、簡単な入力。
	両手で持ち、親指で簡単な操作から中程度の操作まで行う	•左下隅と右下隅に配置したオブジェクトは、すばやく操作できます。 •親指が固定されているため、タッチが正確になります。 •画面の中央には届きにくくなります。 •画面の中央をタッチするには、持ち換える必要があります。 •読み取り、閲覧、簡単な入力、ゲーム。
	デバイスをテーブルや膝の上に置き、両手で簡単な操作から中程度の操作まで行う	•画面の下部に配置したオブジェクトは、すばやく操作できます。 •下の隅は手や手首でふさがれる場合があります。 •指を伸ばす必要が少なくなるため、タッチが正確になります。 •読み取り、閲覧、メール、大量の入力。

	操作するしないに関係なく、デバイスをテーブルや台の上に置く	•画面の下部に配置したオブジェクトは、すばやく操作できます。 •画面の上部をタッチすると、コンテンツが見えなくなります。 •画面の上部をタッチすると、置かれているデバイスがぐらつく場合があります。 •画面から離れて操作するため、読みやすさと正確さが損なわれます。 •ターゲットのサイズを大きくすると、読みやすさと正確さが増します。 •映画、音楽の視聴。

Windows Phone の対話操作と操作性

このトピックでは、レイアウトがアプリの操作性に与える影響について説明します。検索や設定など、その他の UI 要素についても、アプリの操作性との関連で説明します。

コントロールや対話操作について確認する前に、次のことを行ってください。

- 「最良の Windows Phone アプリを設計する」でアプリの概念化戦略について確認する。
- 「コントロール」で、標準的なシステム コントロールの設計ガイドラインを確認する。

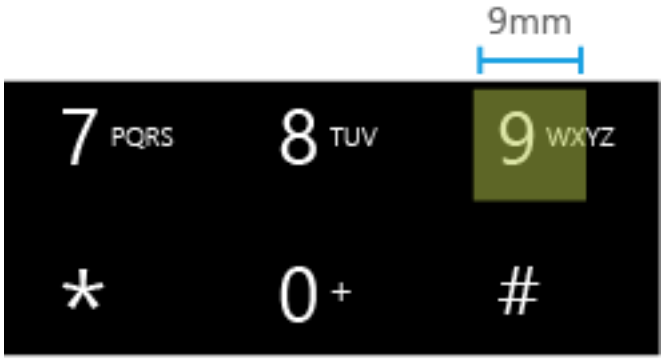
タッチ ターゲットとテキスト

アプリでは、適切なサイズのタッチ ターゲットを表示する必要があります。ユーザーは、自分のタップによってコントロールが動作し、アプリを操作できたというフィードバックを受け取る必要があります。そのため、Windows Phone にはタッチ ターゲットとテキストの使用に関していくつかの要件があります。

サイズの要件

タッチの設計とは、サイズ、間隔、見た目の間で複雑なバランスを取ることです。このバランスをうまく取ると、"ターゲット把握の困難さを示す指標" が軽減されます。つまり、コントロールが押しやすくなります。

大規模なユーザー テストの結果から、Microsoft のすべてのタッチ プラットフォームで、一辺 9 mm の正方形が理想的なタッチ ターゲットのサイズであることがわかっています。



タッチ UI アセットでは一辺 9mm 以上の正方形が理想的なタッチ ターゲット

ヒット ターゲットの高さを低くすることが許可される場合、ターゲットの最小サイズは 7 mm です。このような場合は、ビジュアル資産の幅を広くすることをお勧めします。たとえば、リスト項目やメニュー項目の幅を広くします。

ターゲットの最小サイズは 7mm

9 mm の要件について

タッチ ターゲットの推奨サイズは一辺 9 mm 以上の正方形です。タスクで使われることが多いコントロールには、このサイズを使ってください。

スペースの制約が厳しい場合は、最小サイズである 7 mm のタッチ ターゲットを使用できますが、幅を広くすることが前提です。

9 mm という数字は、数百時間に及ぶユーザー テストから決定されたもので、独立したタスクと連続的なタスクの両方で誤操作の平均発生率 (合計タップ数に対する誤ったタップ数の割合) が最も低いことを表した値です。9 mm を最小タッチ ターゲット サイズにすることで、誤操作の発生率をわずか 1.6% に抑えることができます。

タッチ ターゲットの最小サイズは 7 mm です。このタッチ ターゲットは、使用頻度の低いコントロール、または十分に幅の広い (15 mm 以上) コントロールに対して、設計上の高さ制限がきわめて厳しいときに限って使います。

外見上の最小サイズ

タッチ可能な項目の外見上の最小サイズは、4.2 mm 以上にする必要があります。これより小さくなると、ユーザーはその項目をタッチ対象から完全に除外します。このサイズを使うのは、小さなビジュアル資産が必要なときに限定してください。ターゲットは 10 ～ 15 mm 以上にすることができます。

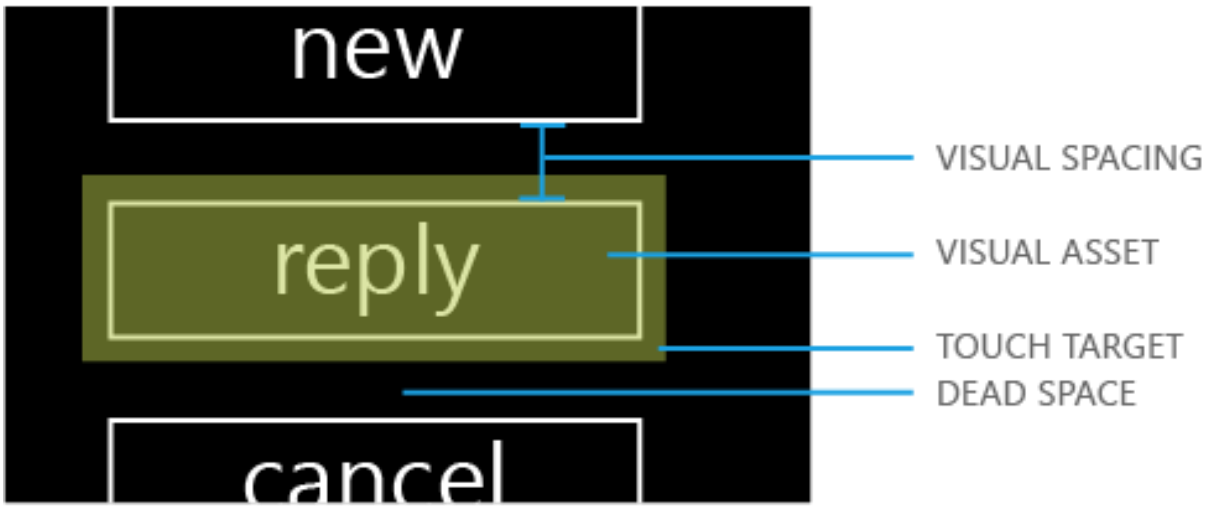
ヒント: ほとんどの場合は、外見上のサイズはタッチ ターゲット サイズと同じにしてください。押すときに見やすくなるように、コントロールの間は間隔を空けます。

ミスタップの結果を評価する

コントロールのターゲット サイズを決定するときは、必ずタスクのコンテキストを把握し、誤操作の発生率を評価します。誤操作が重大な結果につながるコントロールは、ターゲットを大きくする必要があります。動きのあるコントロールも、ターゲットを大きくする必要があります。

たとえば、システムのダイヤラーはタッチ ターゲットが非常に大きくなっています。これはミスタップが間違い電話という重大な結果につながるためです。誤操作がもう少し許容されやすい操作の場合は、ターゲットを小さくすることができます。

次の図に、タッチ ターゲットの各要素を示します。



上のタイル図では、次の点に注目します。

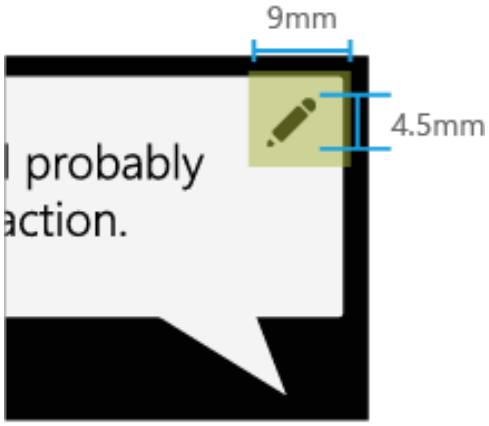
- **外見上の間隔:** コントロールやボタンの間隔を空けると、より快適にタッチできるようになります。
- **ビジュアル資産:** ユーザーに表示されるアイコン/コントロール/テキストの実際のサイズです。
- **タッチ ターゲット:** 上の図でビジュアル資産の周囲に表示されている緑色の境界線です。タッチ ターゲットは、ビジュアル資産と同じかそれ以上のサイズにすることはできますが、ビジュアル資産より小さくすることはできません。共通のコントロール スタイル ガイドでは、緑色の線で示されます。
- **デッド スペース:** タッチ可能な 2 つの項目の間にある、何も操作が行われない領域です。

見栄えのするタイルの作成について詳しくは、「タイルとバッジのガイドライン」をご覧ください。

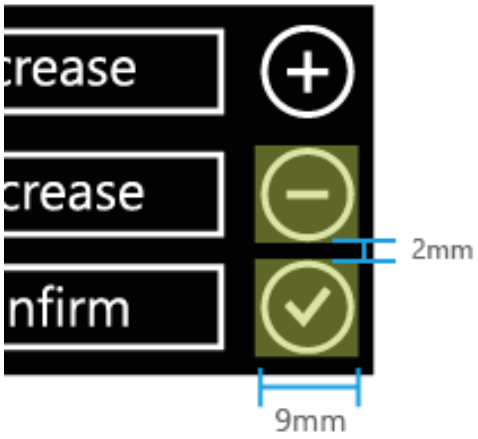
小さなターゲットの使用

スペースとタスクの制約によって適切なタッチ ターゲットの使用が難しい場合は、タッチ ターゲットの次の属性を変更することで操作感をどの程度向上できるかを検討してください。

- 形状
- 位置
- 使用頻度
- タスクのコンテキスト
- 外見上のデザイン (パディング/間隔)
- 誤操作による結果



ターゲット サイズはビジュアル資産よりも大きくします。ビジュアル資産のサイズがターゲットよりも小さくなる場合は、資産のサイズを 4.5 mm より小さくしないでください。



隣接するビジュアル資産の間は間隔を空けます。この場合は、ターゲットをヒットしやすくするために、隣接する資産の間に間隔 (2 mm 以上) を空けてヒット ターゲットのサイズを調整してください。

資産の周囲に外見上のパディングを作成します。外見上のパディングを設定して安全領域を作ることで、小さなターゲットをヒットしやすくすることができます。これにより、ターゲットをヒットするときの感覚上の難しさが軽減します。

UI 要素の適切なサイズを定義する際は、要素とその要素に伴うタスクの重要性を考慮してください。メールのチェックのように日常的なタスクは、ユーザーがそれほど注意を払う必要はありません。特別なタスクにはより注意を向ける必要があります。

小さなコントロール間の間隔

小さな要素がある場合は、間隔を空けることでターゲット サイズを調整できます。間隔は、距離の近い小さなコントロールを調整するために使います。距離の近い小さなコントロールの間は間隔を空けるようにしてください。

重要な注意: 実際のコントロールのサイズに関係なく、9 mm の最小タッチ ターゲット サイズを確保できるだけの間隔を空けてください。

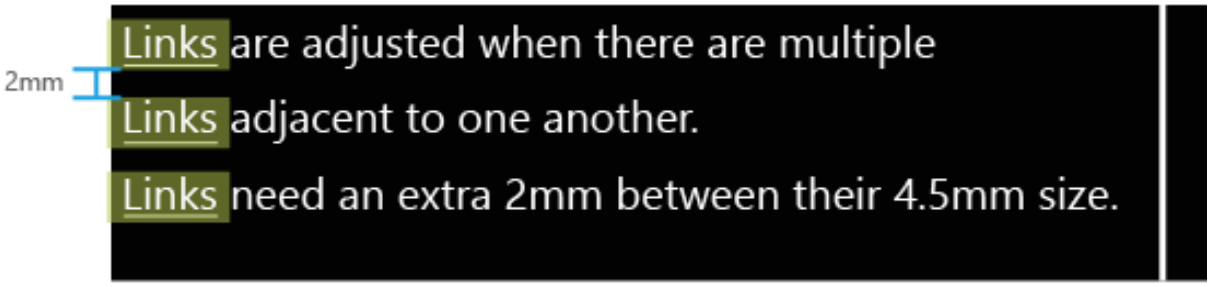
たとえば、4 mm のチェック ボックス コントロールがあるとします。タッチ ターゲットは、サイズが 7 ～ 9 mm になるように 3 ～ 5 mm 大きくします。さらに誤操作を減らすには、チェック ボックス コントロールと隣接する 9 mm のターゲット (262 dpi デバイスでは約 90 ピクセル) との間に間隔を空けます。デバイス上でイメージのサイズをノギスで測ってください。ピクセル数ではなくサイズが重要です。

例外

スクリーン キーボードやハイパーリンク コントロールなど一部のコントロールには、タッチの精度を向上させるためのアルゴリズムを使います。

キーボードやハイパーリンクの一覧のような例外では、ターゲットをヒットしやすくするために、ヒット ターゲットのサイズ変更アルゴリズムやズームなどの訂正メカニズムを使う必要があります。ほとんどの場合、ターゲットの高さの方が幅よりも重要ですが、例外もあります (たとえば、ソフト キーボードのキーでは、幅がターゲットの把握に影響する可能性があります)。高さと幅の少なくとも一方は理想的なターゲット サイズになるようにしてください。

いずれの条件も満たさない場合は、小さなヒット ターゲットの周囲に隣接するヒット ターゲットが多くならないようにしてください。



最小テキスト サイズ

文字要素のサイズと配置は、画面レイアウトを構成するうえで非常に重要です。モダン デザインにはクロム要素がないことが基本であり、これは、テキストや文字が非常に重要な要素であることを意味します。

ヒント: フォントのラベルとプロンプトが読みやすく、適切な間隔になるようにしてください。Windows Phone の最小フォント サイズは 15 ポイントです。

通常は、コンテンツのレイアウトを作り上げるために、中心的な要素に対して境界線やフレームなどの追加的な要素が使われます。さまざまなフォント サイズ、色、またはスタイル

を使うことにより、画面上で必要とされる階層を作り、最も重要なタスクとその次に重要なタスクをユーザーが簡単に識別できるようにします。

カスタム コントロールの作成

アプリのコンテンツがどのように合致しているか、表示されるかを検討している間は、カスタム コントロールを独自に作成する前に、標準的なシステム コントロールの設計ガイドラインを「コントロール」で確認してください。

カスタム コントロールは、機能、タスク、操作を実行または理解しやすくするためだけに作成します。

カスタム コントロールを操作する際のモデルとして、標準的なシステム コントロールを使うことを検討してください。コンテンツの表示中に使われるコントロールは、あまり目立たないようにします。全画面表示の間は、コントロールはフェード アウトする必要があります。

ボタン

Windows Phone プラットフォーム全体で一貫性のあるエクスペリエンスを提供するには、ボタンを配置する際に一般的な構造に従うことが重要です。これにより、ユーザーが移動しやすい一貫性のある簡潔な構造が実現します。

また、ハードウェアとソフトウェアによって自由に使えるものが何かを理解しておくことも重要です。まず、すべての Windows Phone には、戻る、スタート、検索という 3 つの操作ボタンがあります。これらのボタンがシステム全体でどのように使われているかを理解することで、UI 内やアプリの流れの中で発生する問題を軽減できます。

[ホーム] ボタンとバック スタック

ユーザー インターフェイスに [ホーム] ボタンを配置することは、典型的なナビゲーション モデルから逸脱します。プラットフォーム上で革新的なことを行うのは必ずしも悪いことではありませんが、プラットフォームの想定される対話操作モデルを変更すると、ユーザーの混乱を招くおそれがあります。この標準の例外は、アプリがディープ リンクから起動され、バック スタックがない場合です。この場合は、トップ ページに戻る手段が必要になります。

さらに、[ホーム] ボタンをアプリに実装すると、もう 1 つ、アプリにとってさらに深刻な意味を持つ問題が生じる場合があります。それは、ユーザーが [ホーム] ボタンとハードウェアの [戻る] ボタンの両方を使って移動しているときに無限 (またはほぼ無限) ループに入ってしまうという状況が気付かないうちに作り出されてしまうというものです。このループはさらに悪化して、ユーザーがあるアプリで [戻る] ボタンを使って、お客様のアプリに戻ろうとすると、さらに別のアプリに移動する可能性があります。これらの問題の影響をアプリが受けないようにしてください。

すべてのアプリでは対話のフローが異なり、複雑さが異なる場合もあります。ただし、アプリのアーキテクチャはできるだけ浅くし、リストやピボットを活用してユーザーが少ない手順で最初の画面に戻り、また、そこから 1 つ前に起動していたアプリに戻れるようにしてください。自信がない場合は、プラットフォームに実装済みの一般的な要素とナビゲーション構造に似せてみてください。これにより、ユーザーを混乱させる可能性は低くなります。

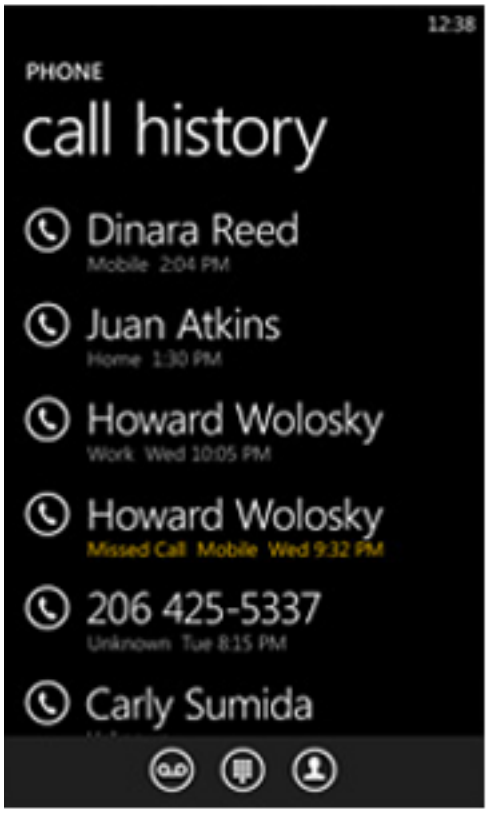
[戻る] ボタンと [閉じる] ボタン

アプリ UI 内には、[戻る] ボタンと [閉じる] ボタンを配置しないでください。Windows Phone では、すべてのデバイスに物理的な戻るボタンが付いているため、アプリのナビゲーションを簡潔明瞭にすることができます。

アプリ内に表示された不適切な [戻る] ボタン と [閉じるボタン]

フローティング ボタン

フローティング ボタンを使うと、ナビゲーションに一貫性がなくなり、混乱します。ページにコマンドを追加する最適な方法では、アプリ バーを使います。アプリ バーについて詳しくは、「基本的なグラフィックス、視覚的なインジケーター、通知 (Windows Phone ストア アプリ)」をご覧ください。



CORRECT

INCORRECT

すべてのコマンドをアプリ バーに配置できない場合は、それらを UI の中で一貫性があるように配置してください。配置を変更可能すると、コンテンツの参照を妨げる可能性があります。また、無計画に配置されたアイコンは、ユーザーから対話要素と認識されない可能性もあります。

Windows Phone の一般的なコントロールのライブラリを使うと、一貫性を保って UI の中にコマンド UI を実装できます。プラットフォーム全体で共通する対話操作を実現するには、「Windows Phone 用の設計のリソース」で推奨されているレイアウトに従ってください。

検索の使用

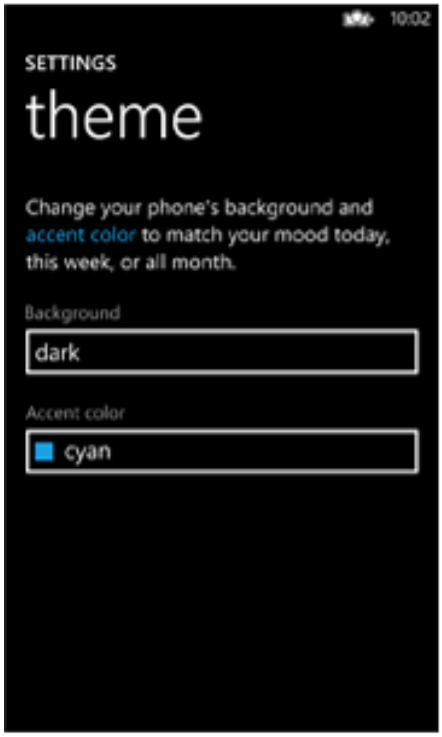
検索は、すべての Windows Phone のハードウェアとソフトウェアに組み込まれています。開発者はハードウェアの検索ボタンの動作を変更することはできません。詳しくは、「初めての Windows Phone」の「検索ボタン」をご覧ください。

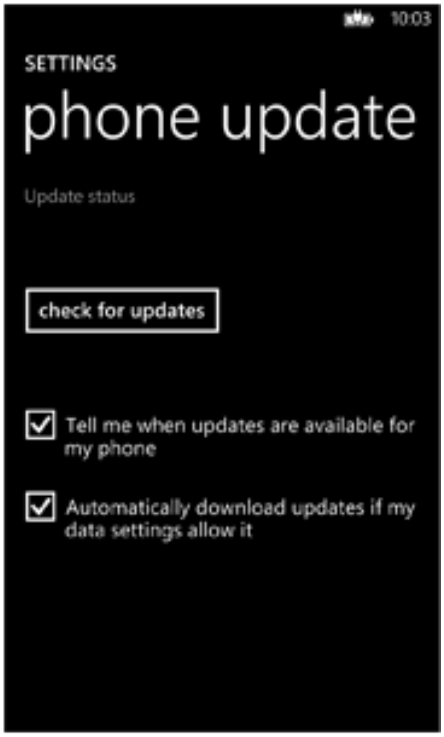
標準的な方法で設定を提示する

Windows Phone では、アプリの設定はそのアプリ自体の中に実装されます。開発者は、システムとアプリケーションのシステム設定内のアプリの設定にはアクセスできません。

ヒント: システム設定のオプションをよく理解し、各種のユーザー設定が UI やアプリの動作に与える影響を考慮してください。たとえば、Web サービスに接続するアプリを作る場合は、ユーザーが電話を機内モードにしたときのアプリの動作について考慮します。

ユーザー選択可能な設定が複数存在するアプリの場合、開発者はアプリ内に設定ページを作成し、そのレイアウトと動作についてはシステムとアプリケーションのシステム設定をモデルにする必要があります。





アプリの設定ページはシステムの設定ページに合わせて作成します。

アプリの設定に対する変更はすぐに適用されます。そのため、[完了]、[OK] などの確認ダイアログは不要です。場合によっては、変更が直ちに行われても、進行中のイベントが完了するか新しいイベントが発生するまでユーザーに変更が発生したというフィードバックが行われないことがあります。フィードバックの提供について詳しくは、「Windows Phone のアニメーション、モーション、出力」をご覧ください。

アプリの設定を簡潔、明瞭に保つことを設計の目標にする必要があります。複雑で、複数のページにわたり、複数のレベルを持つアプリの設定は、ユーザーを不愉快にさせたり、別のアプリに移動してしまったと混乱させたりするおそれがあります。

重要な注意: 3 ページ以上にわたるアプリの設定は作成しないようにしてください。

また、以下の点にも注意してください。

- 複数の画面を必要とする設定では、ソフト キーボードが表示されているときにコンテキストが失われないように、画面の半分が重なって表示されるようにしてください。
- 元に戻すことができないタスクの場合は、必ずユーザーが取り消せるようにしてください。テキスト入力が一例です。
- データを上書きまたは削除する操作や元に戻すことができない操作には、[キャンセル] ボタンを用意してください。
- [適用] ボタンと [キャンセル] ボタンのある追加の画面を使う場合は、これらのボタンをクリックすると、それに関連付けられた操作が実行され、メインの設定画面に戻ります。
- ネットワークを通じてデータを取得するアプリには、データの使用を無効にするオプションを用意してください。

設定コントロール パネルの見出しの一貫性を維持するために、設定ページの見出しは次のようにしてください。

アプリケーションの設定

<コントロール パネル名/アプリ名>

アプリ内広告

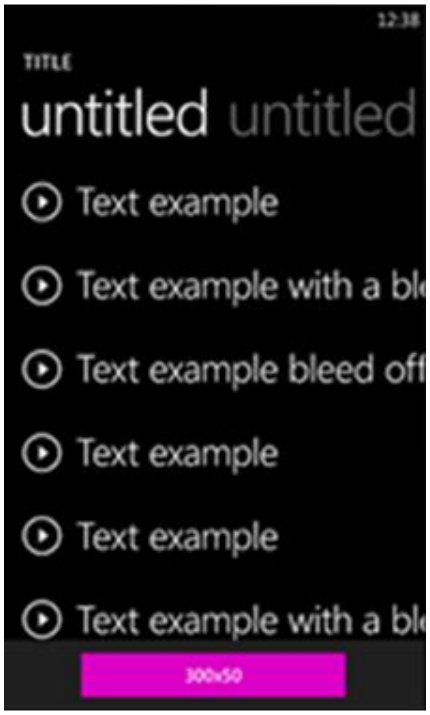
Windows Phone における広告ユニットに求められる最低限の品質に関する基本的なガイドラインは次のとおりです。このガイドラインを確認する際は、広告も Windows Phone のデザインの原則に従う必要があることに留意してください。

表示する広告がない場合は、AdControl 自体が非表示になります。この場合に対処するために、アプリには空いたスペースを利用するためのコードを含めてください。

推奨される広告ユニット サイズ

AdControl の既定かつ推奨されるサイズは、480 x 80 ピクセルです。これは、Windows Phone における広告の推奨サイズです。

広告ユニットを pubCenter のフォーマット サイズである 300 x 50 ピクセルで作った場合でも、ユーザー エクスペリエンスを向上させるために AdControl のサイズは 480 x 80 ピクセルに設定してください。広告ユニットのフォーマットが小さい場合は、pubCenter 内で設定された広告ユニット サイズを使って、AdControl 内の領域の中央に配置されます。たとえば、300 x 50 ピクセルの広告は、AdControl の中央に広告ユニットの中央が位置するように、300 x 50 ピクセルで表示されます。この構成を次の図に示します。

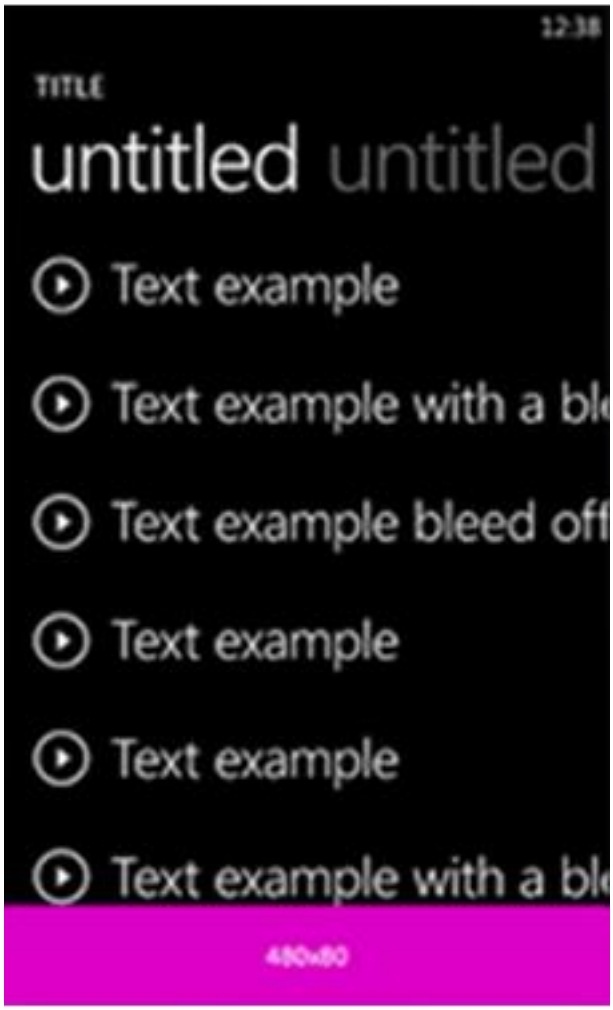


広告コントロールを中央に配置

ヒント: 300 x 50 ピクセルという AdControl のサイズは、下位互換性を維持するために用意されています。将来的には廃止される可能性があります。後で広告ユニットを変更する手間を省くため、新しい広告を作る際は AdControl の標準サイズ 480 x 80 ピクセルを使うことを検討してください。

広告の配置

AdControl は、画面の一番上または一番下に配置します。お勧めの位置は、現在のビューの一番上または一番下です。



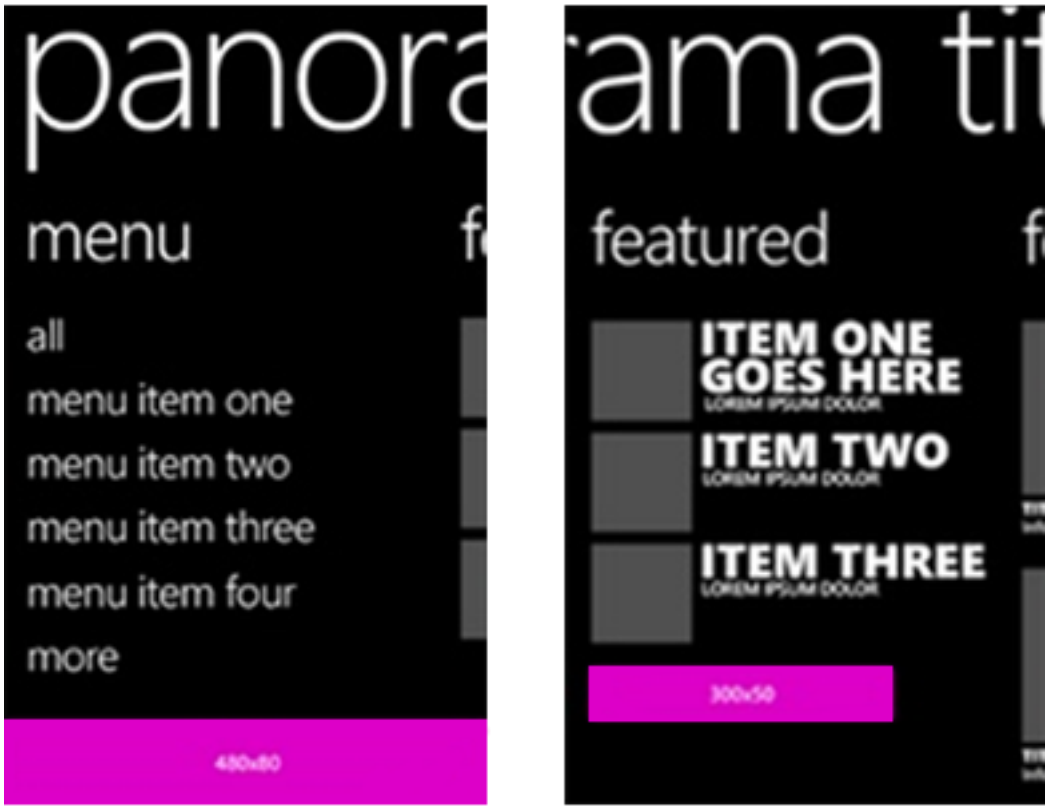
フルサイズの AdControl

スクロール ビューアー

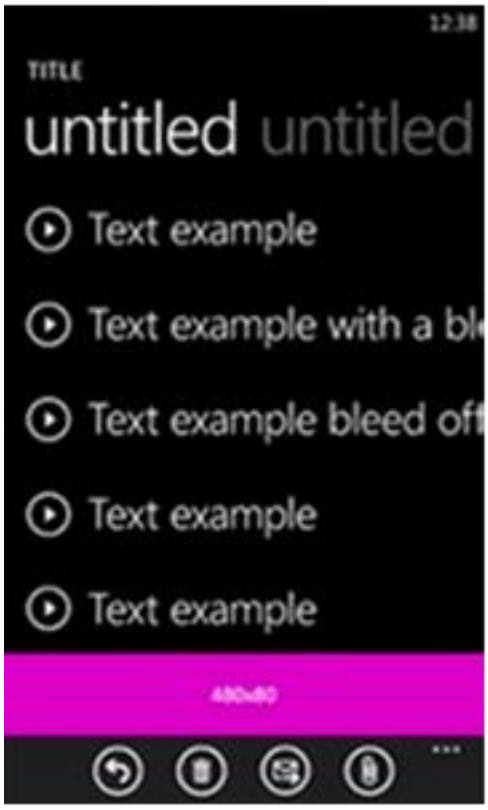
スクロール ビューアー内に含まれる広告は、ユーザーがコンテンツをスクロールすると、ページ内に表示されたり、表示されなくなったりします。ユーザーがスクロールしても広告が動かないようにするには、AdControl をスクロール ビューアーの外部に配置します。スクロール ビューアーは、画面の一番上または一番下に配置します。

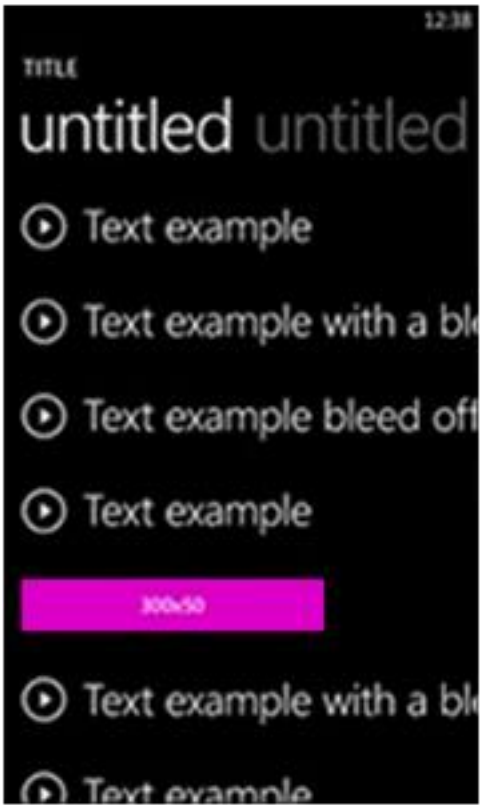
ハブ コントロールまたはピボット コントロールでの広告の使用

ハブ コントロール内のセクション (またはピボット コントロール内の項目) に含まれる広告は、ユーザーが隣接するセクションや項目に移動すると表示されなくなります。広告がすべてのセクションや項目で表示されたままにするには、次の左側の図に示すように、AdControl をコントロールの外に配置します。セクションや項目ごとに異なる広告を表示するには、次の右側の図に示すように、各セクションまたは項目内に、それぞれ異なる広告に対応する AdControl の新しいインスタンスを作成します。



ハブ コントロール内の広告





ピボット コントロール内の広告

色

システムのテーマの色を使います。テーマの色を変更する場合は、AdControl の境界線とテキストが見やすい色を選んでください。

タッチパッド操作 (Windows ストア アプリ)

Windows ストア アプリの設計はタッチ入力用に最適化し、標準で提供されるタッチパッドのサポートを利用します。

ユーザーがタッチパッドで操作できる Windows ストア アプリを設計します。 タッチパッドは、間接的なマルチタッチ入力と、マウスのようなポインティング デバイスの精密入力を組み合わせたものです。この組み合わせにより、タッチパッドは Windows 8.1 のタッチに最適化された UI にも、デスクトップ環境での生産性アプリのより小さいターゲットにも適しています。

このトピックでは、タッチパッド操作の設計時の考慮事項について説明します。

Windows 8.1 タッチパッド言語

Windows 8.1 には、システム内で一貫して使われるタッチパッド操作の簡単なセットが用意されています。

用語	説明
2 本指でのタップによる右クリック	2 本の指で同時にタップすると、グローバル コマンドが含まれているアプリ バーが表示されます。要素上でタップすると、その要素が選択され、状況依存のコマンドが含まれているアプリ バーが表示されます。 **注** 選択またはアプリ バーのコマンドが適切な UI 動作ではない場合は、2 本指でのタップによりコンテキスト メニューが表示されます。ただし、すべてのコマンド動作にアプリ バーを使うことを強くお勧めします。

2 本指でのスライドによるパン	スライドは主にパン操作に使われますが、移動、描画、筆記などの操作に使うこともできます。
ピンチとストレッチによるズーム	タッチパッドでのピンチとストレッチは、サイズ変更とセマンティック ズームに使われます。
システム コマンドのためのエッジの操作	タッチパッドの右端 (右から左のレイアウトでは左端) からスワイプすると、システム コマンドが含まれているチャームが表示されます。 タッチパッドの左端 (右から左のレイアウトでは右端) からスワイプすると、実行中のアプリの一覧が表示されます。
左と右のクリック ゾーン	マウス デバイスの左ボタンと右ボタンの機能をエミュレートします。
ホバーによる説明の表示	要素にホバーすると、詳しい情報や説明を伝えるビジュアル効果 (ツールチップなど) が表示されます。操作は確約されません。
1 本指でのタップによるプライマリ操作	1 本指を使って要素をタップすると、プライマリ操作 (アプリの起動、コマンドの実行など) が呼び出されます。
1 本指でのプレスとスライドによる移動	要素をドラッグします。
1 本指でのプレスとスライドによるテキストの選択	選択可能なテキスト内を押してスライドし、選択します。単語を選択するには、ダブルタップします。

ビジュアルなフィードバックのガイドライン

- 移動イベントまたはホバー イベントを通じてタッチパッド カーソルが検出されたら、マウス固有の UI を表示して、要素によって公開されている機能を示します。タッチパッド カーソルが一定の期間動かされなかった場合や、ユーザーがタッチ操作を始めた場合は、タッチパッド UI を徐々に非表示にします。 これにより、UI の簡潔さが保たれます。
- ホバーのフィードバックにカーソルを使わないでください。要素によるフィードバックで十分です。
- 静的テキストなど、要素で対話操作がサポートされていない場合は、ビジュアルなフィードバックを返さないでください。
- タッチパッド操作ではフォーカス用の四角形を使わないでください。これはキーボード操作専用です。
- 同じ入力対象を表すすべての要素に対してビジュアルなフィードバックを同時に表示します。

マウス操作 (Windows ストア アプリ)

Windows ストア アプリの設計はタッチ入力用に最適化し、標準で提供されるマウスのサポートを利用します。

ユーザーがマウスで操作できる Windows ストア アプリを設計し、構築します。

マウス入力は、ポイントとクリックの正確さが求められるユーザー操作に最適です。この固有の正確さは、Windows の UI でも当然サポートされていますが、タッチの本来の不正確さに合わせて最適化されています。

マウス入力とタッチ入力が異なるのは、タッチでは画面上の UI 要素に対する物理的なジェスチャ —(スワイプ、スライド、ドラッグ、回転など) を通じて、それらのオブジェクトへの直接の操作をエミュレートする機能があることです。

このトピックでは、マウス操作の設計時の考慮事項について説明します。

マウスの言語

Windows には、システム内で一貫して使われるマウス操作の簡単なセットが用意されています。

用語	説明
ホバーによる説明の表示	要素にホバーすると、詳しい情報や説明を伝える視覚効果 (ツールチップなど) が表示されます。操作は確約されません。
左クリックによるプライマリ操作	要素の左クリックにより、プライマリ操作 (アプリの起動、コマンドの実行など) が呼び出されます。

スクロールによるビューの変更	スクロール バーを表示し、コンテンツ領域内で上下左右に移動します。スクロール バーのクリック、またはマウス ホイールの回転により、スクロールできます。スクロール バーは、コンテンツ領域内の現在のビューの位置を示します (タッチによるパンでも同様の UI が表示されます)。
右クリックによる選択とコマンド	右クリックして、ナビゲーション バー (使用できる場合) と、グローバル コマンドを含むアプリ バーを表示します。要素を右クリックして選択し、その要素に対応する状況依存のコマンドを備えたアプリ バーを表示します。 **注** 選択またはアプリ バーのコマンドが適切な UI 動作ではない場合は、右クリックによりコンテキスト メニューが表示されます。ただし、すべてのコマンド動作にアプリ バーを使うことを強くお勧めします。
ズームの UI コマンド	アプリ バーに UI コマンドを表示するか (+、- など)、Ctrl キーを押しながらマウス ホイールを回転させて、ズームのためのピンチ ジェスチャーとストレッチ ジェスチャーをエミュレートします。
回転の UI コマンド	アプリ バーに UI コマンドを表示するか、Ctrl キーと Shift キーを押しながらマウス ホイールを回転させて、回転のための回転ジェスチャーをエミュレートします。画面全体を回転させるには、デバイスを回転させます。
左クリックとドラッグによる移動	要素を左クリックしてドラッグし、移動します。
左クリックとドラッグによるテキストの選択	選択可能なテキスト内を左クリックしてドラッグし、選択します。単語を選択するには、ダブルクリックします。

マウス カーソルを隅とエッジに移動したときにシステム コマンドを表示	左から右方向の画面で右上隅または右下隅 (右から左方向のレイアウトでは左隅) にマウスを動かすと、システム コマンドが含まれているチャームが表示されます。 左から右方向のレイアウトで左上隅 (右から左へのレイアウトでは右上隅) にマウスを移動すると、最近使ったアプリのサムネイルが表示されます。左クリックするか、クリックしてドラッグし、実行中のアプリを切り替えます。ドラッグしてアプリをスナップします (この操作がサポートされている画面解像度の場合)。 左から右方向のレイアウトで左下隅 (右から左方向のレイアウトでは右下隅) にマウスを移動すると、スタート画面のサムネイルが表示されます。 左クリックして画面の上端から下端にドラッグすると、現在のアプリが閉じます。 左クリックして画面の上端から下端か、左端または右端へスライドすると、現在のアプリがスライドした側にスナップされます。

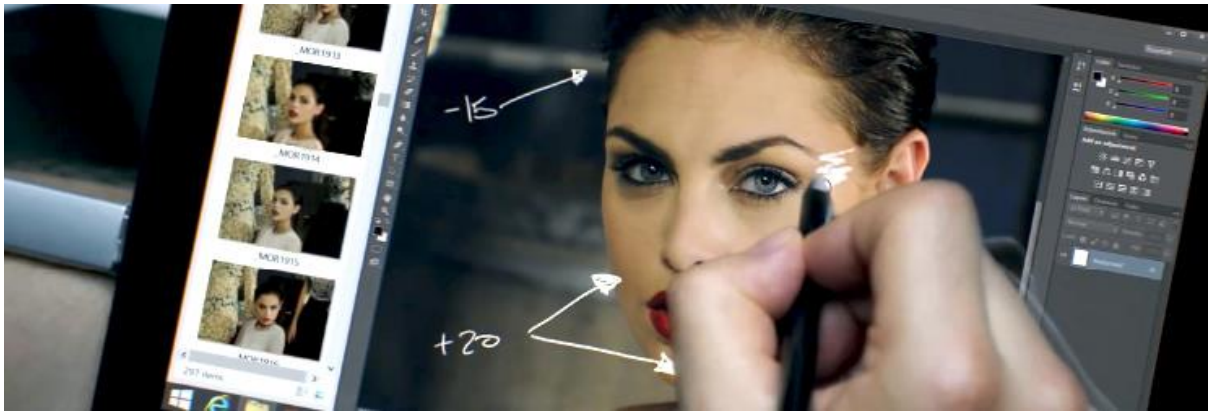
ビジュアルなフィードバックのガイドライン

- 移動イベントまたはホバー イベントを通じてマウスが検出されたら、マウス固有の UI を表示して、要素によって公開されている機能を示します。マウスが一定の期間動かされなかった場合や、ユーザーがタッチ操作を始めた場合は、マウス UI を徐々に非表示にします。 これにより、UI の簡潔さが保たれます。
- ホバーのフィードバックにカーソルを使わないでください。要素によるフィードバックで十分です (以下の「カーソル」をご覧ください)。
- 静的テキストなど、要素で対話操作がサポートされていない場合は、ビジュアルなフィードバックを返さないでください。
- マウス操作ではフォーカス用の四角形を使わないでください。これはキーボード操作専用です。
- 同じ入力対象を表すすべての要素に対してビジュアルなフィードバックを同時に表示します。

- パン、回転、ズームなど、タッチ ベースの操作をエミュレートするためのボタンを提供します (+、- など)。

ペン操作 (Windows ストア アプリ)

Windows ストア アプリの設計はタッチ入力用に最適化し、標準で提供されるペンのサポートを利用します。



このトピックでは、ペン操作の設計時の考慮事項について説明します。

ペン操作

ペンを使うことで、ユーザーが手書きノート、描画、コメントの書き込み操作を行うことができる Windows ストア アプリを設計できます。

ペンは、ポインティング デバイスとして使うことができます。より興味深い用途は、ペンをデジタル インクに関連付けられた描画デバイスとして使うことです。 Windows のインク プラットフォームでペン デバイスを使うと、自然な形で手書きノート、描画、コメントを作れます。

ユーザーが書いたり描画したりするときのペンの空間移動のキャプチャに加えて、アプリで筆圧、形状、色、不透明度などの情報を収集して、紙の上でペン、鉛筆、ブラシを使っているときに近いユーザー エクスペリエンスを実現することもできます。

ペン入力が持つ固有の正確さは、UI でも当然サポートされていますが、タッチの本来の不正確さに合わせて最適化されています。

ビジュアルなフィードバックに関するガイダンスについては、「ビジュアルなフィードバックのガイドライン」をご覧ください。

キーボード操作 (Windows ストア アプリ)

ユーザーがハードウェア キーボード、スクリーン キーボード、またはタッチ キーボードを通じて操作できる Windows ストア アプリを設計します。

このトピックでは、キーボード操作の設計時の考慮事項について説明します。

キーボード操作

キーボード入力は、Windows ストア アプリのユーザー操作エクスペリエンスの中でも重要な要素です。キーボードは、特定の障碍のあるユーザーや、キーボードを使った方がアプリを効率よく操作できると考えるユーザーにとって欠かせません。

適切に設計されたキーボード UI はソフトウェアのアクセシビリティの重要な要素であり、視覚に障碍のあるユーザーや、特定の運動障碍のあるユーザーがアプリ内を移動したり、その機能を操作したりするのに使います。このようなユーザーはマウスを操作できないため、代わりにさまざまな支援技術 (キーボード強化ツール、スクリーン キーボード、スクリーン拡大機能、スクリーン リーダー、音声入力ユーティリティなど) が不可欠である可能性があります。

最も一般的な種類のキーボードは、デバイスに物理的に接続されている外付けのハードウェア キーボードです。ハードウェア キーボードに加えて、Windows には 2 種類のソフトウェア キーボードが用意されています。

- スクリーン キーボードは、物理的なキーボードの代わりに使うことができるビジュアルなソフトウェア キーボードです。タッチ、マウス、ペン/スタイラス、またはその他のポインティング デバイス (タッチ スクリーンは必須ではありません) を

通じてデータを入力します。スクリーン キーボードは、物理的なキーボードが存在しないシステムや、運動障碍により一般的な物理入力デバイスを使うことができないユーザーのために用意されています。スクリーン キーボードは、ハードウェア キーボードの機能のすべて、または少なくともほとんどをエミュレートします。

- タッチ キーボードは、タッチ入力でのテキスト入力に使われる、ビジュアルなソフトウェア キーボードです。タッチ キーボードはテキスト入力専用であり (ハードウェア キーボードをエミュレートしません)、テキスト フィールドや編集可能なテキスト コントロールにフォーカスがあるときにだけ表示されるので、スクリーン キーボードの代わりになるものではありません。

  注 スクリーン キーボードの方がタッチ キーボードより優先され、スクリーン キーボードが表示されている場合はタッチ キーボードは表示されません。

  次に、タッチ キーボードの例を示します。最初の画像は既定のレイアウトであり、2 つ目の画像は親指レイアウトです (一部の言語では利用できません)。

ユーザーが、アプリでサポートされているすべてのタスクをハードウェア キーボードまたはスクリーン キーボードだけで実行できるようにしてください。

注 タッチ キーボードはテキスト入力にのみ使い、アプリ コマンドやシステム コマンドの入力には使いません。

キーボード コマンド パターン

信頼性の高い一貫したユーザー エクスペリエンスを実現するために、次の標準キーボード コマンド パターンをお勧めします。次の 3 つの表に、よく使われるキーボード コマンドを示します。

ナビゲーション コマンド

操作	キーコマンド
戻る	Alt + ←、または特殊キーボードの戻るボタン
進む	Alt + →
上へ	Alt + ↑
キャンセルまたは現在のモードの終了	Esc
一覧の項目間の移動	方向キー (←、→、↑、↓)
次の項目一覧へジャンプ	Ctrl + ←
セマンティック ズーム	Ctrl + 正符号 (+) または Ctrl + マイナス記号 (-)
コレクション内の名前付き項目にジャンプ	項目の名前を入力します
次のページ	PageUp、PageDown、または Space
次のタブ	Ctrl + Tab
前のタブ	Ctrl + Shift + Tab
アプリ バーを開く	Windows+Z
アクティブ化、または項目を開く	Enter
選択	Space
連続して選択	Shift + 方向キー
すべてを選択	Ctrl + A

一般的なコマンド

操作	キーコマンド
項目をピン留めする	Ctrl + Shift + 数字 1
保存	Ctrl + S
検索	Ctrl + F

印刷	Ctrl + P
コピー	Ctrl + C
切り取り	Ctrl + X
新規項目	Ctrl + N
貼り付け	Ctrl + V
開く	Ctrl + O
アドレスを開く (たとえば、Internet Explorer で URL を開く)	Ctrl + L または Alt + D

メディア ナビゲーション コマンド

操作	キーコマンド
再生/一時停止	Ctrl + P
次の項目	Ctrl + F
前の項目	Ctrl + B

キーボードのフォーカスとナビゲーションのガイドライン

- アプリの起動時に、ユーザーが最初に直感的に操作する (または、その可能性が最も高い) 要素に、最初のキーボード フォーカスを設定します。
- 含まれる要素が少ないアプリでは、Tab キーを使ってコレクションまたはリスト内で次の個別要素にキーボード フォーカスを移動できるようにします。多数の要素が含まれるアプリでは、Tab キーを使ってキーボード フォーカスを要素のグループ間で移動できるようにします。
- Shift + Tab のキー コマンドを使って、逆方向にキーボード フォーカスを移動できるようにします。
- 方向キーを使って違和感なくアプリのコンテンツ内を前後に移動できるようにします。ただし、リーディング アプリは例外です。読み取りのためにコンテンツを提示する場合、方向キーでページ単位にコンテンツをスクロールできるようにします。

- Enter キーを使って、コレクション内または一覧内のすべての選択項目をアクティブ化できるようにします。
- Ctrl キーをしばらく押したままにすることで自動選択を一時的に無効にできるようにします。自動選択が無効になっていると、Space キーを押すことで項目を明示的に選択または選択解除できます。
- キーボードを使ってサーフェスを開いたとき、その新しいサーフェスの最初の要素にキーボード フォーカスが表示されるようにします。
- Esc キーを押してサーフェスを閉じることができるようにします。サーフェスが開かれた元の場所にキーボード フォーカスの表示を戻します。

ビジュアルなフィードバックのガイドライン

- キーボード操作でのみフォーカス用の四角形を使います。ユーザーがタッチ操作を始めたら、キーボードの UI を徐々にフェード アウトします。 これにより、UI の簡潔さが保たれます。
- 静的テキストなど、要素で対話操作がサポートされていない場合は、ビジュアルなフィードバックを返さないでください。
- 同じ入力対象を表すすべての要素に対してビジュアルなフィードバックを同時に表示します。
- パン、回転、ズームなど、タッチ ベースの操作をエミュレートするためのツールチップとして画面ボタンを提供します (+、- など)。

基本的なグラフィックス、ビジュアル、インジケーター、および通知 (Windows Phone ストア アプリ)

美しさと直感的な操作が同じことを意味するのであれば、モバイル アプリには美しさが不可欠です。Windows Phone では、タイル、スプラッシュ スクリーン、アイコン、コントロール、およびナビゲーションのビジュアル要素が、アプリ内の関連タスク、優先順位、または操作に着目しながら、斬新かつ魅力的な方法で情報を表示します。アプリには、ライブ タイルのカスタム デザインやアニメーション アイコンだけでなく、アプリの読み込み中にそのアプリについてユーザーに紹介するスプラッシュ スクリーン画像が必要になります。このセクションでは、こうしたビジュアル インジケーターについて説明します。

むやみにグラフィックを使うのはやめましょう。視覚に訴えるには、コンテンツと文字体裁を使います。装飾だけを目的としてビジュアル要素を使わないでください。

モバイル プラットフォームでは、シンプルさが最大の魅力になります。アートの使用が適切な場所では、高品質のオリジナル グラフィック アート、ブランド、または写真を使います。アプリが必要とする基準以上のアートを使い、簡潔で明瞭、かつ知的な外観を実現してください。

タイル

タイルはアプリまたはそのコンテンツのわかりやすいショートカットで、電話のスタート画面で設定できます。スタート画面に表示するタイルは、アプリのタイルをビューに固定することでカスタマイズできます。ライブ タイルは、アプリ情報またはアプリに関連する情報を表示できる点でアイコンよりも優れています。たとえば、天気のアプリのタイルには温度を動的に表示できます。この機能によりアプリがより便利になるので、この機能を積極的に使うことを強くお勧めします。

優れたタイルは、アプリに関する情報を伝達し、ブランドの特徴をよく表します。また、画面上の他のタイルの中でひときわすばらしく見えます。タイルは、システム フォントを使うオプションのカウンターを表示したり、提供されたタイルの背景画像を更新したり、固定されたサイズおよび色でシステム フォントを使うオプションのタイルを表示したりすることで、ユーザーに情報を伝達します。カウンター、背景画像、およびタイルの更新は、タイ

ル通知サービスを使って制御されます。カウンターのアクセント カラーは必ず、ユーザーが選択したアクセント カラーです。カウンターの表示は省略可能です。

Windows Phone デバイスには、Microsoft とその携帯電話会社およびハードウェア メーカーのパートナーによって既に組み込まれているタイルがいくつか付属しています。これらのタイルは、常に Windows Phone 独自のエクスペリエンスを示します。

詳しくは、「タイルとバッジのガイドライン」をご覧ください。

タイル アートが重要

複数のタイル画像を使う場合は、視覚的な一貫性を確保し、テーマまたはスタイルがわかるようにします。色、フォント、フォントの色、またはカウンターの表示サイズは変更できません。

タイル画像またはタイトルが組み込まれていないアプリについては、一般的なシステム定義アイコンとプロジェクト名が表示されます。アプリ開発におけるデザインの予算が厳しい場合は、リーズナブルな料金でアイコンを購入できる Web サイトが多数存在するのでご利用ください。自分でデザインするにしても、手数料を払って使うにしても、購入するにしても、タイルは常にシンプルにします。アイコンのジオメトリは単純にして、細部にこだわるのは控えましょう。可能であれば、既によく知られているメタファーを利用します。

注 タイル通知は慎重に使ってください。使いすぎるとバッテリー残量が少なくなります。

避けるべきデザイン

- 3D 文字体裁
- グラデーション、面取り、または影が適用されたアイコンや背景
- 角の丸め
- 黒または白の背景。ダーク テーマまたはライト テーマを設定すると、黒または白のタイルの背景は見えなくなります。
- あいまいで説明的でないアイコンの使用
- 透明な背景とカラフルな画像

タイルの背景色

タイルには 2 つの主要要素があります。1 つは前景に表示されるアイコンまたはロゴ、もう 1 つは四角形のカラーの背景です。この 2 つがお互いを引き立たせる必要があります。

ブランドを表す背景色を選択し、前景のアイコンを見やすく、または読みやすくします。次の図は、このガイドラインに従っている 3 つの例を示しています。

Phone Company、Adatum、Margie's Travel

黒または白の背景色は避けてください。黒または白のタイルの背景は、ダーク テーマまたはライト テーマを使うと見えなくなるからです。

タイルに黒の背景色を不適切に使った例

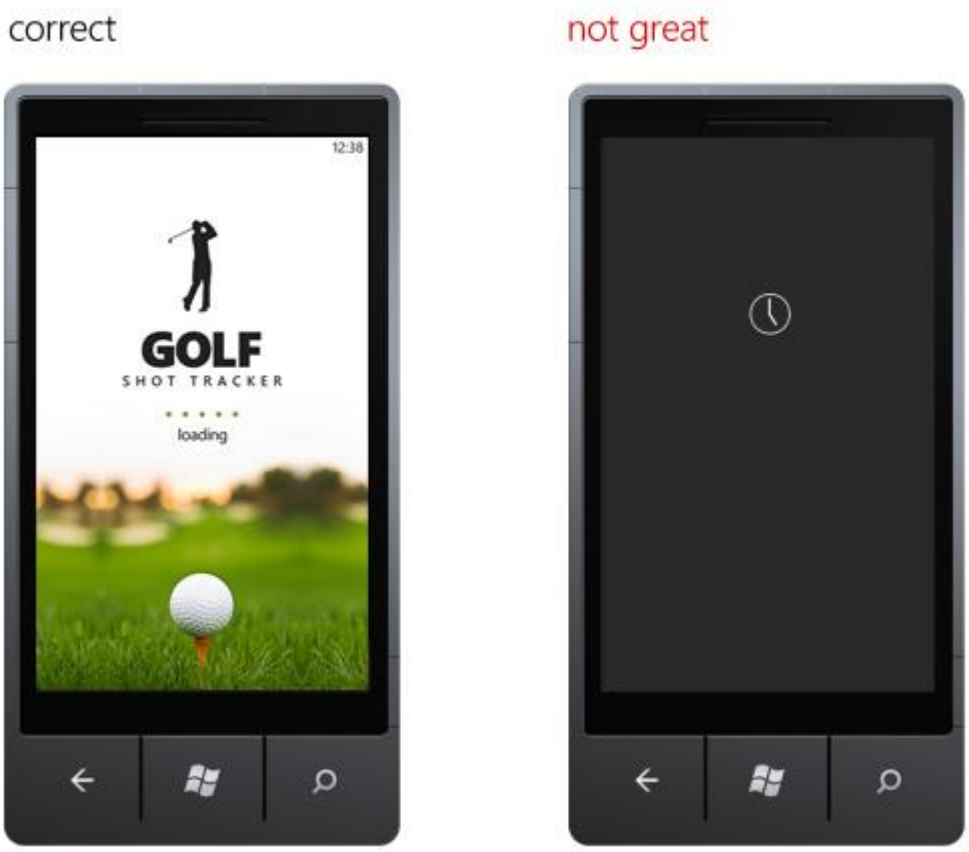
タイルの背景を電話の UI と同じテーマ色にしたい場合は、タイルの背景を透明にします。この場合は、次の操作を行うことが非常に重要です。

- タイルのみを透明にします。アプリで送信する他のアイコンの背景は透明にしません。
- 前景のアイコンは白にします。前景のアイコンに色が付いていると、そのアイコンは見えなくなるか、電話のテーマ色の一部と重なって見えにくくなります。以下の図に例を示します。

前景のアイコンを白にしなかった場合の問題

スプラッシュ スクリーン

読み込みに時間がかかるアプリは多数あります。この機会に、スプラッシュ スクリーンでアプリをユーザーに紹介しましょう。スプラッシュ スクリーンは数秒しか表示されないので、ユーザーが注意を向ける必要がある、または読むのに時間がかかるテキストは配置しないようにすることをお勧めします。代わりに、グラフィックと共にユーザーがアプリを順に眺められるようにします。優れたスプラッシュ スクリーンはアプリのグラフィック広告であるため、その目的に沿った色やグラフィックを使います。



適切なスプラッシュ スクリーンと不適切なスプラッシュ スクリーン

アプリのオープニング ビューを計画するときは、次の点に注意してください。

- アプリで読み込みページまたは紹介ページが使われている場合、そのページはバック スタックに含めないでください。つまり、ユーザーが [戻る] ボタンを押したら、これらのページはスキップされなければなりません。
- [戻る] ボタンとアプリの初期画面に関しては、いくつかの認定要件があります。詳しくは、Windows Phone の技術認定要件に関するページをご覧ください。

ビジュアル インジケーター

Windows Phone では、3 種類のインジケーターでタスクの進行状況、スクロール ビュー、シグナルの強さ、バッテリー残量、およびその他の重要な情報を示します。

進行状況インジケーター

進行状況インジケーターは、アプリ内のアクティビティに関連する動作や一連のイベントを示します。これはステータス バーに組み込まれたシステム コントロールで、複数のアプリ ページにわたって表示できます。ステータス バーについて詳しくは、このトピックで後述する「ステータス バー」をご覧ください。

注 標準の進行状況バーと似ていますが、ステータス バーの進行状況インジケーターは確定されているか不確定かのどちらかです。確定された進行状況インジケーターには先端と終端があります。不確定の進行状況インジケーターの場合は、タスクが完了するまで終端が示されません。

このインジケーターを、コンテンツのダウンロードなどのタスクには確定モードで使い、リモート接続などのタスクには不確定モードで使うことができます。

ユーザーにタスクの進行状況を表示する方法について詳しくは、「プログレス コントロール (Windows Phone の ProgressBar コントロールのデザイン ガイドライン)」をご覧ください。

スクロール インジケーター

ーテキストや画像の長いセクションなどー、画面上のコンテンツが表示されているページに収まらない場合、ユーザーがパンまたはフリックすると、ページがスクロールします。ユーザーがスクロールすると、垂直方向のスクロールの場合は右側に、水平方向のスクロールの場合は下部に、スクロール インジケーターが表示されます。これらのスクロール バーは、コンテンツがページの高さまたは幅よりも大きいこと、およびページ内での現在の位置を表します。ページのスクロールが終了すると、スクロール インジケーターは 1 秒後に消えます。

ユーザーがこのインジケーターを操作することはできません。これは、インジケーターの下にあるコンテンツのオーバーレイで、その主な役割は、ページのサイズに関するヒントをユーザーに提供することです。

スクロール ビューアーと呼ばれるパンおよびズーム可能なコントロールを使って、アプリのスクロールを実装できます。スクロール ビューアーは通常、画面いっぱいに表示され、

画面サイズより大きいコンテンツを表示します。これにより、画面の境界を超える場合があるコンテンツ領域内を移動できます。

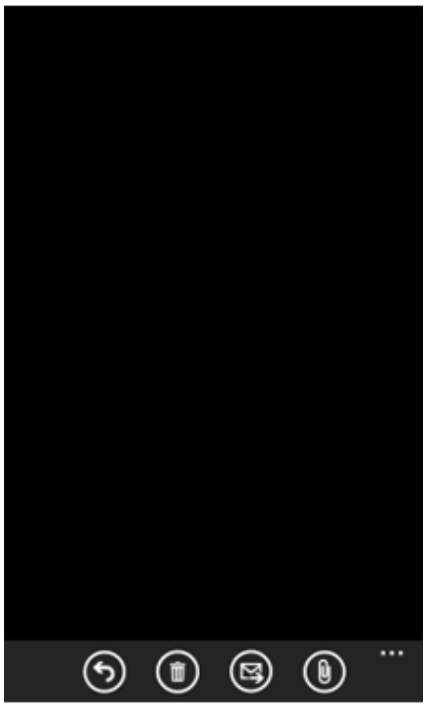
ステータス バー

ステータス バーは、システム レベルのステータス情報を、アプリのワークスペースの予約された部分にシンプルかつ明確に表示するインジケーター バーです。このバーは自動的に更新され、さまざまな通知を表示し、システム レベルのステータスをユーザーが常に把握できるようにします。

詳しくは、「初めての Windows Phone (Windows Phone 早わかり)」の「Status Bar (ステータス バー)」をご覧ください。

アプリ バー

アプリ バーは、最も一般的なアプリ タスクおよびビューを最大 4 つアイコン ボタンとして表示できる場所です。このバーには、アイコン ボタンとテキスト ヒントが表示されるビューと、アプリ バー メニューと呼ばれるコンテキスト メニューがオプションで示されます。このアプリ バー メニューは、ユーザーが連続するドットのビジュアル インジケーターをタップするか、アプリ バーを上にフリックするとアクティブになります。



独自のメニュー システムを作成するのではなく、アプリ バーを使ってください。これは、デバイス上のすべてのアプリで一貫したユーザー エクスペリエンスを作成するのに役立ちます。アプリ バーでは、メニューのアニメーションおよび回転がサポートされています。

アプリ バーは、マークアップだけのアプリのページに追加できます。

設計時の考慮事項:

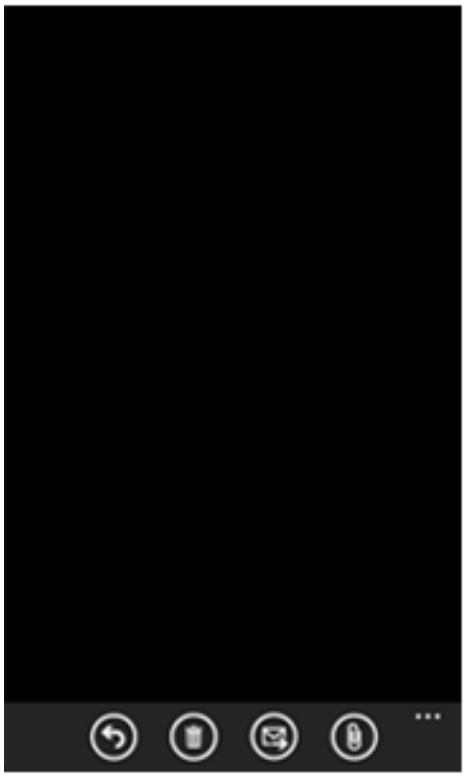
- 上書きする切実な理由がない限り、ユーザー定義のシステム テーマ カラーを使います。ボタン アイコンでユーザー設定の色を使うと、ボタン アイコンの表示品質に影響を及ぼしたり、メニュー アニメーションに予期しない視覚効果をもたらしたりする場合があります。また、デバイスによってはバッテリー残量が少なくなることもあります。
- アプリ バーの不透明度は細かく調整できますが、不透明度の値には 0、0.5、および 1 を使うことをお勧めします。
- アプリ バーは、常に操作ボタン ([戻る]、[スタート]、および [検索]) と同じディスプレイの端に表示され、縦向き、横向きどちらでも画面の幅全体に拡張されます。アイコン ボタンは電話の向きに合わせて回転します。
- アプリ バーの縦向きでの高さと横向きでの幅は固定で、変更できません。バーは表示または非表示に設定することができます。

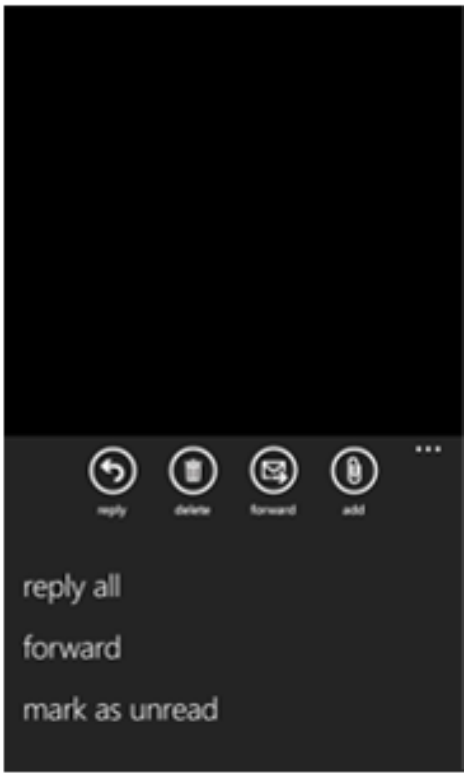
アプリ バーのローカライズ方法に関する説明および例については、「Windows Phone のローカライズしたアプリケーションの構築方法」をご覧ください。

アプリ バーのメニュー

アプリ バー メニューはオプションです。このメニューを使うと、ユーザーがアプリ バーの特定のタスクにアクセスできます。アプリ バー メニューには、あまり使わないタスクを配置します。

このメニューは、ユーザーが連続するドットのビジュアル インジケーターをタップするか、アプリ バーを上にフリックするとアクティブになります。ビューを閉じるには、メニュー領域外またはドット上をタップするか、[戻る] ボタンを使うか、メニュー項目またはアプリ バー アイコンを選択します。





設計時の考慮事項:

- メニューがスクロールされないようにするには、表示するメニュー項目数を 5 個に制限します。
- アプリ バー メニュー項目のテキストが長すぎる場合、そのテキストは画面の外にはみ出ます。メニュー項目のテキストの推奨される最大文字数は 14 ～ 20 文字です。繰り返しになりますが、このスペースでは少ない文字数の方がより効果的になります。
- 表示されるメニュー項目がない場合は、アイコンのテキスト ヒントのみが表示されます。
- ユーザーがアクションを実行するまでアプリ バー メニューは画面上に残ります。

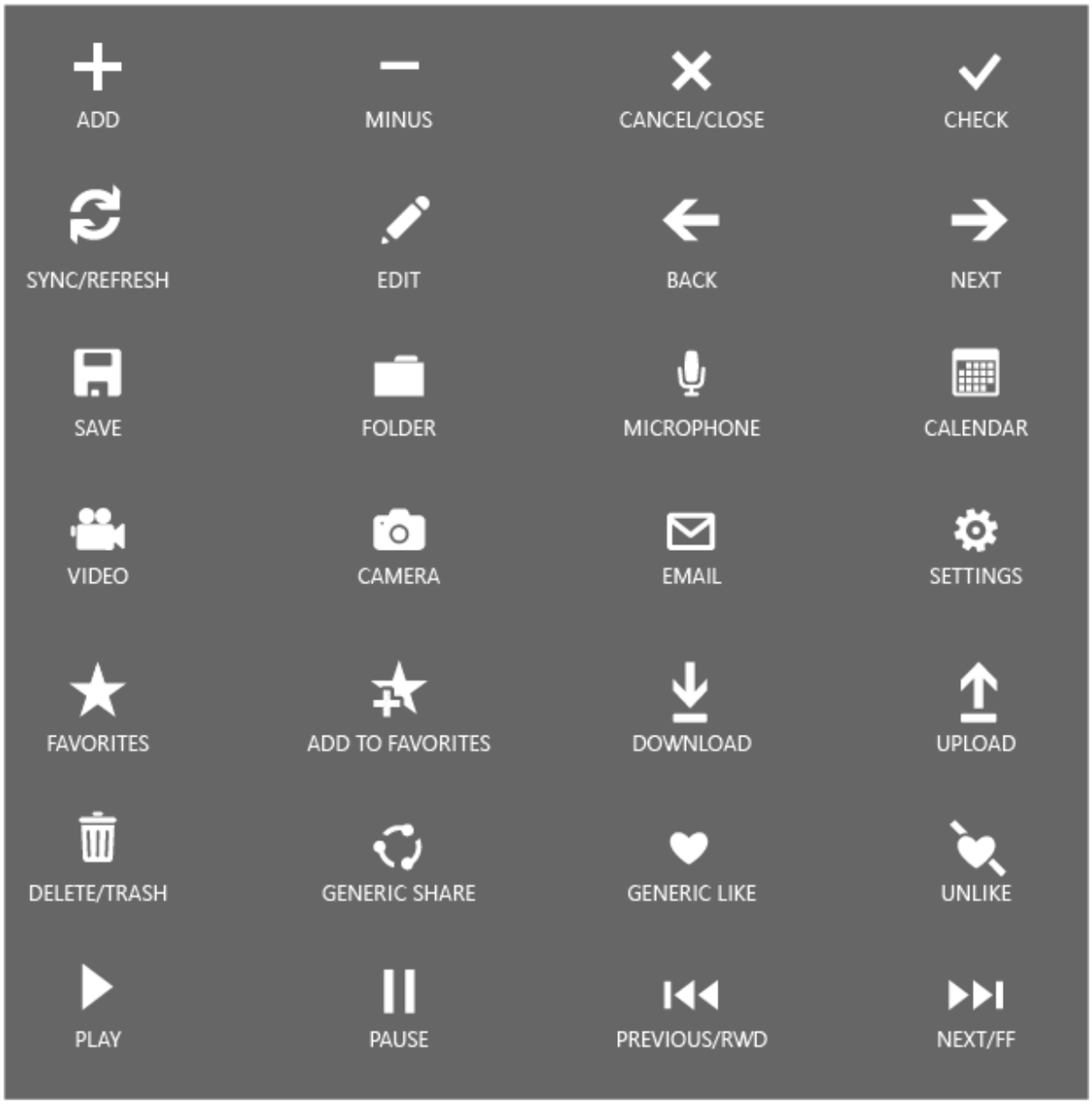
アプリ バー アイコン

アプリ バー アイコンは明確でわかりやすくなければなりません。また、ユーザーによく知られている実際のメタファーを利用します。優れたアイコンは単純なジオメトリで、細かな表現は抑えられています。

ボタンにはアイコンおよびテキスト ヒントを付けます。テキスト ヒントは、ボタンの動作を簡単に説明するものです。詳しく説明する必要はありません。たとえば、画像を水平にフリップするボタンの場合は、"水平にフリップ" ではなく "フリップ" で十分です。

注 ボタンにはアイコンおよびテキスト ヒントを付けます。

アプリ バー ボタンは有効または無効な状態で表示できます。たとえば、読み取り専用の項目では [削除] ボタンを無効状態で表示するなどが可能です。



アプリでよく使う主要な操作に対してアイコン ボタンを使います。使うという理由だけで多くのアイコンを採用しないでください。

使用上の考慮事項:

- 操作によっては、アイコンで意味を明確に伝達するのは困難です。この場合は、これらの操作をアプリ バー メニューに配置してください。
- ユーザーがアプリ バー メニューを表示すると、アプリ バー アイコンのテキスト ヒントが表示されます。
- ページ スタックを後ろに移動する [戻る] ボタンに対して、アイコン ボタンを使わないでください。すべての Windows Phone デバイスに、後ろへの移動で必ず使われる専用のハードウェア [戻る] ボタンが必要です。
- アプリ バーを回転したときに、意味が明確に伝わるアイコンを選択します。アプリ バーは、画面の向きの変化を自動的に処理します。デバイスが横向きになっている場合は、画面の側面に垂直にメニューが表示されます。アイコン ボタンはユーザーに対して垂直になるように回転しますが、リスト内のアイコンの順序が変更されることはありません。このような場合、特に、類似した、または対称を成す 2 つのアイコンでは、アイコンの意味が紛らわしくないかどうかを検討してください。

設計時の考慮事項:

- アイコン画像のサイズは、76 x 76 ピクセルとします。
- アイコン画像では、アルファ チャネルを使って白の前景色と透明の背景を使う必要があります。ボタン用の白の前景グラフィックは、円と重ならないように、画像の中心の 41 x 41 領域の正方形に合わせる必要があります。
- 推奨サイズ以外の画像はスケーリングされ、アプリ バー アイコンの全体的な画像品質よりも低くなる可能性があります。
- 各アイコン ボタンに表示される円はアプリ バーによって描画され、ソース画像には含まれません。

サンプル アイコン資産

一連の PNG 形式のアプリ バー アイコン資産 (暗いアイコンと明るいアイコン) が Windows Phone SDK の一部としてインストールされます。アプリ バーでは白のアイコン (Dark フォルダー内のアイコン) のみを使ってください。これらのアイテムを探すには、次の場所に移動します。

C:\Program Files (x86)\Microsoft SDKs\Windows Phone\<version>\Icons\Dark

通知

アプリ開発については、Windows Notification Service が、回復機能を持つ専用の永続チャネルをクラウド サービスに提供するために設計され、通知をモバイル デバイスにプッシュします。

クラウド サービスは、プッシュ通知をデバイスに送信する必要があるときに、通知要求を Windows Notification Service に送信します。Windows Notification Service はその通知を、タイル通知、トースト通知、または直接通知としてアプリまたはデバイスにルーティングします。

プッシュ通知を表示するには、タイル通知、トースト通知、直接通知の 3 つの方法があります。

タイル通知

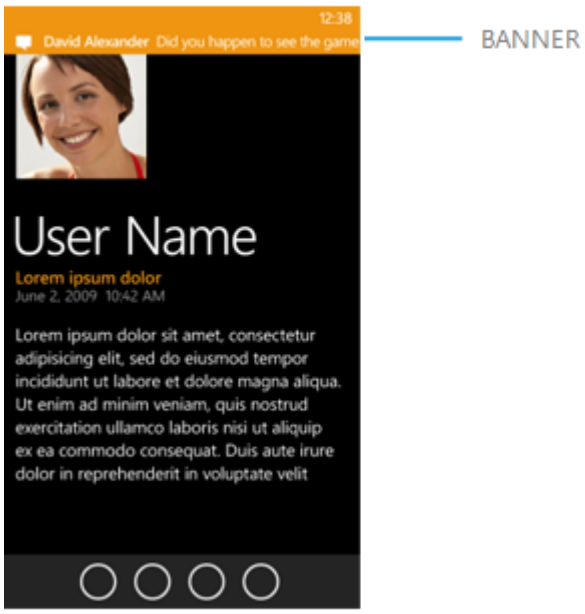
タイル通知は、発生した可能性のある変更またはイベントを、ユーザーのワークフローを中断せずにユーザーに通知します。この通知は、スタート タイルに表示されます。タイル通知を使うと、数、背景画像、タイトルなどのプロパティをタイルで動的に設定できます。

通知のみが目的の場合にタイル通知を使います。

トースト通知

Web サービスでは、ユーザー操作を要求するトースト通知と呼ばれる特別なプッシュ通知を生成できます。トースト通知は画面上部に不透明なバーとしてアクセント カラーで表示され、左隅にはアプリ アイコンの縮小版が表示されます。また、太字のタイトルと通常文字のサブタイトルの 2 つのテキスト フィールドを使うことができます。表示領域に収まらないテキストは切り捨てられます。

バナーが 10 秒表示された後、見えなくなります。バナーをタップすると、そのトースト通知を送信したアプリが起動します。トースト通知はシステム全体の通知ですが、ユーザーのワークフローを中断しないか、解決のための介入を要求します。たとえば、ユーザーがテキスト メッセージまたはインスタント メッセージを受信したときにこの通知が表示されます。



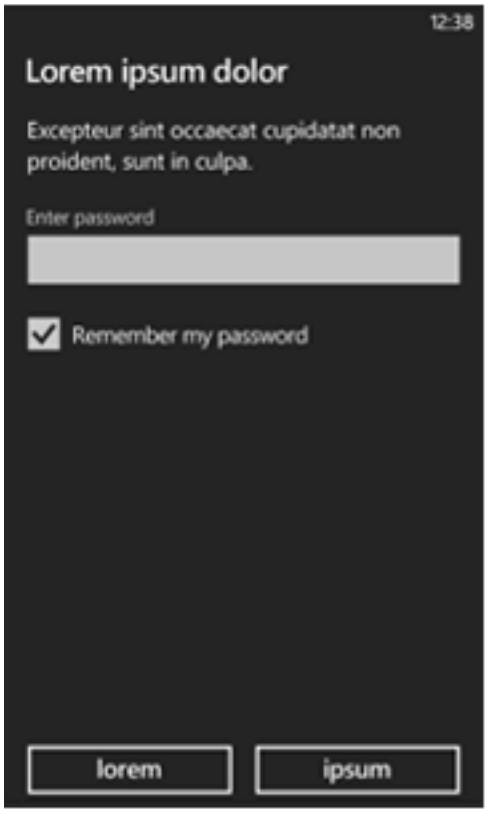
バナー

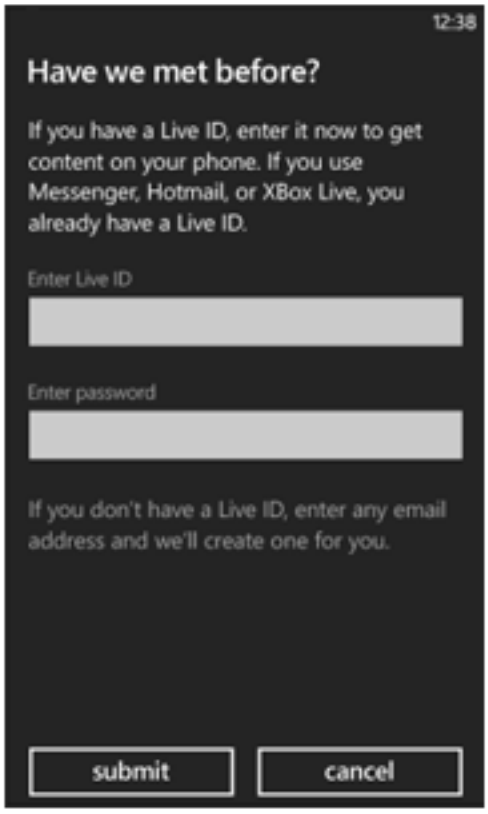
操作の要求を通知する場合にトースト通知を使います。この通知の例としては、インスタント メッセージ クライアントまたはピアツーピア指向のアプリを介して生成される通知が挙げられます。ただし、この通知の使用は控えめにしてください。すべてのアプリがトースト通知にアクセスでき、バナーが多すぎるとユーザーが不快になるからです。

アプリの既定値では、トースト通知をオフにする必要があります。トースト通知は、ユーザーに個人的に関連し、迅速性がきわめて重要です。一般的に、これらの通知は、SMS アプリケーションなどのピアツーピア通信用に予約されています。

直接通知

アプリ内でのアクションを要求する通知は、アプリによって完全に制御され、そのアプリにのみ影響を及ぼします。このような通知は直接通知と呼ばれます。この通知はアプリ自体で生成できます。また、Web サービスから送信することもできます。システム全体で直接通知を表示する方法はありません。





ユーザー操作を要求するアプリ内の通知に対して直接通知を使います。

ネットワーク停止の場合

- ネットワークに接続されていない場合、アプリは適切な警告を表示する必要があります。機内モードを使うと、警告をテストできます。
- 使えるネットワークがない状態でも、アプリがまだ操作できることを確認します。データが入ってこなくても、アプリは機能しなければなりません。
- ネットワーク停止によって機能または操作が中断される場合は、何に問題があるのかを必ずユーザーに示します。

エンド ユーザー使用許諾契約書の追加

エンド ユーザー使用許諾契約書 (EULA) は、ユーザーが最初にアプリを起動したときにユーザーが遵守することに同意する使用上のガイドラインです。また、策定したユーザーの権利も明記されています。これは、ユーザーがアプリを悪用しないことを保証する開発者とアプリ購入者間の契約です。

ユーザーは、EULA に同意するとアプリを使うことができます。EULA の条項に同意しないユーザーは、アプリの使用が許可されません。

この声明では、コンテンツ、ライセンス、インストール、アクティブ化、検証、インターネット ベースのサービス、または特定の情報の使用に関する特定の条項を含め、保証するアプリの動作や使用について説明します。ここで、アプリの更新またはアプリへのアップグレード提供の計画や、ソフトウェア形式および標準の使用についてユーザーに説明してもよいでしょう。ビジネス関連のアプリの場合は、事業登録されている場所、およびすべての種類の保証を提供するかどうかを示す必要もあります。

UX ガイドライン

ここからは、すでに説明したモダン デザイン、ビジョン、独創的なビジュアルとインタラクティブと UX に含まれていない UX ガイドラインを提示します。UX ガイドラインの詳細は、Windows デベロッパー センターを参照してください。

テキストと入力

ここでは、フォントと文字体裁、アイコン、テキスト ボックス、テキストの形式、テキストのコピーと貼り付けのコマンドについて説明します。さらに、ドキュメント ビューアーやリーダー アプリなどでページの特定のテキストを検索するためのガイドラインも示します。

このセクションの内容

トピック	説明
クリップボード コマンド	クリップボード コマンド (コピー、貼り付け、切り取り) を使うと、ユーザーが使い慣れた方法で、ある場所から別の場所にコンテンツを移動することができます。これらのコマンドは、ユーザーがコンテンツを移動する際に役立ちます。 • 同じアプリ内 • Windows ストア アプリ間 • デスクトップ アプリケーション間 • Windows ストア アプリとデスクトップ アプリケーションの間 Windows 8 と Windows 8.1 では、コンテンツの共有など、アプリが情報を交換するための他の方法もサポートしていますが、コピーと貼り付けのコマンドは、Windows の操作環境として想定されている部分として残ります。可能な場合には常に、アプリでこれらのコマンドをサポートしてください。

ページ内検索	このトピックでは、Windows ストア アプリでのページ内検索機能の実装に関するベスト プラクティスについて説明します。
フォント	フォントを選び、フォントのサイズと色を指定するときには、次のガイドラインに従ってください。
フォーム レイアウト	優れたタッチ エクスペリエンスを提供し、画面上の領域の使用を最適化し、Windows ストア アプリでパンとスクロールを最小限にするフォームを設計します。
パスワード ボックス	パスワード ボックスは、プライバシーの目的で入力文字が非表示になるテキスト入力ボックスです。
Segoe UI Symbol アイコンの一覧	このドキュメントでは、Segoe UI Symbol フォントが提供する、アイコンとして使うことができる便利なグリフを示します。
スペル チェック	テキストの入力と編集を行っているときに、スペル チェックは単語を赤い波線で強調表示してユーザーに単語のスペルの間違いを知らせ、それを修正する方法を提供します。
テキスト入力	JavaScript を使った Windows ストア アプリにテキスト入力コントロールを追加するためのガイドライン。
タイポグラフィ	Microsoft デザインのすべての原則で、タイポグラフィの重要性が認識されています。アプリの開発者は、高度なタイポグラフィ機能をサポートする一連のフレームワークを初めて利用できるようになりました。

クリップボード コマンドのガイドライン

クリップボード コマンド (コピー、貼り付け、切り取り) を使うと、ユーザーが使い慣れた方法で、ある場所から別の場所にコンテンツを移動することができます。これらのコマンドは、ユーザーがコンテンツを移動する際に役立ちます。

- 同じアプリ内
- Windows ストア アプリ間
- デスクトップ アプリケーション間
- Windows ストア アプリとデスクトップ アプリケーションの間

Windows 8 では、共有など、アプリが情報を交換するための他の方法もサポートしていますが、コピーと貼り付けのコマンドは、Windows の操作環境として想定されている部分として残ります。可能な場合には常に、アプリでこれらのコマンドをサポートしてください。

推奨と非推奨

- ドキュメントや画像のサブセットなど、ユーザーが明示的に選択できる編集可能なすべてのコンテンツに対して、コピーと貼り付けをサポートします。
  例：
  - フォト ギャラリー アプリケーション内の画像
  - 電卓の計算結果
  - レストラン検索アプリケーションでのレストランの住所
- ユーザーが他の場所で使いたいと考える可能性があるコンテンツに対して、コピーと貼り付けのコマンドをサポートすることを検討します。
- 権利の管理や、コピーと貼り付けのコマンドの使用に制限があると思われるその他の要因に留意してください。たとえば、閲覧権利制限付きメールをサポートするアプリの場合、そのようなコンテンツの全部または一部をユーザーがコピーすることをポリシーによって制限する場合があります。
- ユーザーがコピーする対象、またはユーザーがコンテンツを貼り付けることができる場所が明確であるか確認します。
- アプリケーション内の編集可能な領域およびキャンバスに対してのみ、貼り付けのサポートを提供します。

- コピーと貼り付けを使うとコンテンツが削除または置換される場合があるため、元に戻すコマンドの実装を検討します。
- コピーと貼り付けをサポートするコントロールが既にある場合は、そのコントロールの実装を使います。コピーと貼り付けの実装を独自にビルドする必要がある場合は、作る操作環境とそれらのコントロールの一貫性を保つようにします。
- コピーもサポートしている場合は、共有のサポートも検討します。
- ユーザーがコンテキスト メニューとアプリ バーのどちらを使ってコピーと貼り付けのコマンドにアクセスするかを決定します。次の場合は、コンテキスト メニューを使います。
  - ハイパーリンクや埋め込み画像など、タップしたまま押さえるジェスチャでのみ選ぶことができる項目たとえば、アプリでユーザーにアドレスを表示し、ユーザーがそのアドレスをコピーできるようにするとします。最適なユーザー エクスペリエンスは、ユーザーがアドレスを右クリックするか、タップしたまま押さえるとアクセスできるアドレスのコピー コマンドを作ることです。このコマンドによってアドレスがクリップボードにコピーされ、そのクリップボードからユーザーは希望するアプリに貼り付けることができます。

  - テキスト選択 (編集可能と読み取り専用の両方)
  - カーソル位置や表のセルなど、対象が明確に定義されている場合の貼り付け操作

  前のガイドラインに該当しない場合は、アプリ バーを使います。次のような例があります。

  - 複数の項目の選択をアプリでサポートする場合
  - ユーザーが画像の一部を選ぶことができる場合
  - スクリーン ショットをキャンバスに貼り付けるなど、貼り付けコマンドの対象が明確な場合
- Microsoft では、クリップボード コマンドのためにキーボード ショートカットを常にサポートすることをお勧めします。

- 明示的に選ぶことができない、またはコンテキスト メニューを使って選ぶことができないコンテンツのコピーはサポートしないでください。
- アプリケーションの主要なコンテンツではないテキストのコピーはサポートしないでください。タイトル、ヘッダー、およびボタンのテキストはコピーする必要はありません。
- クリップボードが空であるか、クリップボードにアプリでサポートされないコンテンツが含まれている場合は、貼り付けコマンドを有効にすることは避けてください。

ページ内検索のガイドライン

このトピックでは、Windows ストア アプリでのページ内検索機能の実装に関するベスト プラクティスについて説明します。

推奨と非推奨

- ページ内検索を実装し、ユーザーが現在のテキスト本文から一致を検索できるようにする。
- アプリ バーを使って、ユーザーがアプリでテキストを検索できるようにする。
- キーボード ショートカット (Ctrl + F) を有効にする。
- 検索チャームを使って現在のテキスト本文のテキストを検索しない。
- 要約ウィンドウを使わない。

その他の使い方のガイダンス

ページ内検索の実装

ページ内検索により、ユーザーは現在のテキスト本文から一致を検索できるようになります。

- ページ内検索を提供する可能性が最も高いアプリの種類は、ドキュメント ビューアーとリーダーです。
- これらのアプリの主な目的は、ユーザーが全画面の表示または読書エクスペリエンスを得られるようにすることです。
- ページ内検索機能は補助的な機能であり、ユーザーが必要に応じて使うために提供されるその他の機能と共にアプリ バーに配置する必要があります。

検索チャームを使うと、Web ページ、ムービー、イベントなどの一連の項目を検索できます。 ユーザーが検索ボックスにクエリを入力すると、関連する結果セットが返されます。 一般に、これらの結果は関連性アルゴリズムを使って並べ替えられます。

新しい Windows UI の Internet Explorer のページ内検索ユーザー エクスペリエンス。

リーダー アプリのページ内検索ユーザー エクスペリエンス。

アプリへのページ内検索の追加に関する推奨事項は次のとおりです。

- アプリ バーを使って、ユーザーがアプリでテキストを検索できるようにする。ページ内検索機能は補助的な機能であり、アプリ バーに配置する必要があります。
  - ページ内検索を提供するアプリでは、必要なすべてのコントロールがアプリ バーにある必要があります。
  - ページ検索以外に多くの機能をアプリに含める場合は、ページ内検索のすべてのコントロールが含まれる別のアプリ バーへのエントリ ポイントとして最上位のアプリ バーに **[検索]** ボタンを追加できます。
- タッチ キーボードの操作時もページ内検索のアプリ バーが表示されるようにする。タッチ キーボードは、ユーザーが入力ボックスをタップすると表示されます。ページ内検索のアプリ バーは、タッチ キーボードに隠れないように上へ移動する必要があります。
- ユーザーが表示を操作しているときもページ内検索を利用できるようにする。ユーザーは、ページ内検索を使いながら表示内のテキストを操作する必要があります。たとえば、ユーザーはテキストを読むために、ドキュメントを拡大表示または縮小表示したり、表示をパンしたりすることがあります。ユーザーがページ内検索を使い始めたら、アプリ バーはページ内検索を終了するための**閉じる**ボタンと共に表示されたままにする必要があります。
- キーボード ショートカット (Ctrl + F) を有効にする。キーボード ショートカット Ctrl + F を実装し、ページ内検索のアプリ バーをユーザーがすぐに呼び出すことができるようにします。

- ページ内検索機能の基本要素を含める。ページ内検索を実装するために必要な UI 要素を次に示します。
  - 入力ボックス
  - [前へ] ボタンと [次へ] ボタン
  - 一致数
  - 閉じる
- 表示で一致した結果が強調表示され、スクロールして次の一致が画面に表示されるようにする。ユーザーは、[前へ] ボタンと [次へ] ボタンの使用、スクロール バーの使用、またはタッチによる直接操作によってドキュメントをすばやく移動できます。
- 検索と置換の機能が基本的なページ内検索機能と共に機能するようにする。検索と置換の機能があるアプリでは、ページ内検索が検索と置換の機能の妨げにならないようにします。
- 要約ウィンドウは繰り返しになるため、ページ内検索機能に含めないでください。
  - ユーザーが探している一致を識別する最も簡単な方法は、ドキュメント内で確認することです。インラインで一致を確認する方が、結果の要約ウィンドウよりもずっと多くのコンテキストが与えられるためです。
  - ユーザーは、**[前へ]** ボタンと **[次へ]** ボタンの使用、スクロール バーの使用、またはタッチによる直接操作によってドキュメントをすばやく移動できます。
  - 要約ウィンドウを含めることにした場合は、次のガイドラインに従ってください。
    - ユーザーがトグル ボタンを使って要約ウィンドウにアクセスできる必要があります。
    - 要約ウィンドウをオンに切り替えた場合、入力ボックスにテキストがあるときは常に表示される必要があります。
    - ユーザーがページ内検索のアプリ バーを閉じた場合、要約ウィンドウは次回ユーザーが入力ボックスにテキストを入力するまで表示されない必要があります。
    - 要約ウィンドウをオフに切り替えた場合、ユーザーがオンの状態に戻すまでは表示されない必要があります。

フォントのガイドライン

フォントのサイズ、太さ、色、トラッキング、間隔を適切に設定すると、外観がきれいに整い、Windows ストア アプリが使いやすくなります。フォントを選び、フォントのサイズと色を指定するときには、次のガイドラインに従ってください。

ガイドラインを守るために役立つスタイル リソースがあります

- JavaScript を使った Windows ストア アプリには、WinJS スタイル シート を使います。これらのスタイル シートでは、このトピックで説明する、推奨される色、太さ、トラッキング値が使われています。
- XAML を使って Windows ストア アプリを作成する場合は、TextBlock や RichTextBlock のスタイリングに関連付けられたテーマ リソースを使って、これらのガイドラインを簡単に実装できます。

Segoe UI Symbol アイコンの一覧については、「Segoe UI Symbol アイコンのガイドライン」をご覧ください。

推奨と非推奨

簡潔な印象を与えるには、タイポグラフィの細部に注意する必要があります。 アプリの UI の開発では、次の規則を適用することをお勧めします。

- 推奨フォントを使います。
  - ボタンや日付選択コントロールなどの UI 要素には、Segoe UI (主な Windows 書体) を使います。Segoe UI では、ラテン、キリル、ギリシャ、アラビア、ヘブライ、アルメニア、グルジアの各アルファベットがサポートされています。
  - メールやチャットのように、ユーザーが読んだり書いたりするテキストには Calibri を使います。Calibri では、ラテン、ギリシャ、キリルの各アルファベットがサポートされています。以下の「その他の使い方のガイダンス」セクションの「テキストの表示と入力用」をご覧ください。
  - 雑誌や RSS リーダーなどのまとまった量のテキストには Cambria を使います。Cambria では、ラテン、ギリシャ、キリルの各アルファベットがサポートされて

います。以下の「その他の使い方のガイダンス」セクションの「テキスト表示用」をご覧ください。

注 他の言語と書記システムでは、別のフォントが使われ、異なるサイズや間隔が必要になる場合があります。アプリをローカライズするときに組み込みのフォントを使う場合は、Windows.Globalization.Fonts API を使って、その言語の正しい UI とドキュメント フォントを特定してください。

- UI 要素では、次のフォント スタイルとサイズを使います。
  - 通常のテキストでは、11 ポイントの Segoe UI Semilight を使います。
  - ボタンのキャプションなどの短いテキスト要素では、最も小さいサイズの、9 ポイントの Segoe UI を使います。
  - 見やすく目立つ必要があっても、1 行に収まるテキスト要素では、20 ポイントの Segoe UI Light を使います。20 ポイントが大きすぎる場合、16 ポイントの Segoe UI Semilight を使います。たとえば、項目がたくさんあるリストを表示する場合は、16 ポイントのフォントの方が適しています。ただし、同じページで 20 ポイントと 16 ポイントの Segoe UI フォントを同時に使わないでください。
  - 1 行が 1 語または 2 語で構成される人目を引く UI 要素では、42 ポイントの Segoe UI Light を使います。

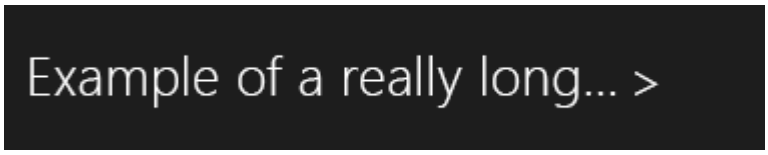
  Segoe UI の太さ、不透明度、文字間隔、Stylistic Set の推奨事項については、「その他の使い方のガイダンス」をご覧ください。

- 必要に応じて、ページ要素を区別できるように、対照的な設定 (太さ、サイズ、位置) を使います。
- 次の不透明度を使います。
  - プライマリ テキストの不透明度は 100% です。
  - セカンダリ テキストの不透明度は 60% です。
  - ホバー状態のテキストの不透明度は 80% です。
  - 押された状態の白黒テキストの不透明度は 40% です。その他の色のテキストの不透明度は 60% です。

さまざまなサイズと不透明度の Segoe UI フォントのサンプルについては、「その他の使い方のガイダンス」の「Segoe UI の書体見本 (type ramp)」をご覧ください。

- キーボード フォーカスは、白と黒が交互に表示されるピクセルの正方形で示します。

- ページ ヘッダーまたはサブヘッダーが長すぎて表示しきれない場合は、省略記号 (U+2026) を使います。3 個のピリオドは使わないでください。また、ヘッダー テキストの末尾と省略記号の間にスペースを挿入しないでください。



- Windows ストア アプリ用のレイアウト ガイドラインに従います。 詳しくは、「アプリのページのレイアウト」をご覧ください。
- 斜体フォントは使いません。

その他の使い方のガイダンス

Segoe UI の定義済みの太さ

Segoe UI は、システムで最も認識しやすいフォントであり、新しい Windows UI と最も関連性が深いフォントです。すべてのユーザー インターフェイス要素で Segoe UI を使います。

Windows ストア アプリ スタイル シートでは、次の 5 つの異なるバージョンの Segoe UI が用意されており、それぞれに文字の太さが違います。

- Segoe UI Light
  太さが 200 の Segoe UI です。
- Segoe UI Semilight
  太さが 300 の Segoe UI です。
- Segoe UI

太さが 400 の標準の Segoe UI です。

- Segoe UI Semibold
  太さが 600 の Segoe UI です。
- Segoe UI Bold
  太さが 700 の Segoe UI です。

これらの定義済みの太さを使うには、カスケード スタイル シート (CSS) で font-family プロパティにいずれかの Segoe UI フォント名を設定します。この例では、Segoe UI Semibold を使います。

```
<p style="font-family: 'Segoe UI Semibold'">Semibold text</p>
```

```
<TextBlock x:Name="semiBold" Text="Semibold font." FontFamily="Segoe UI Semibold"/>
```

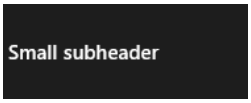
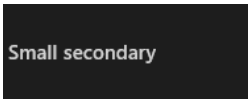
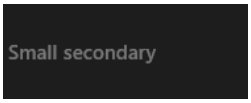
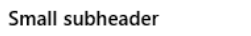
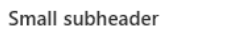
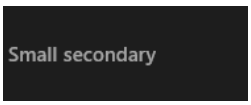
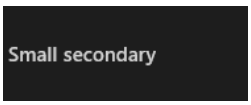
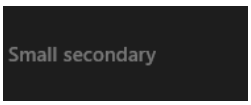
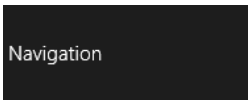
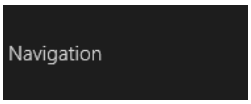
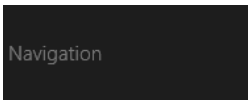
Segoe UI の書体見本 (type ramp)

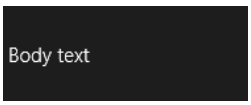
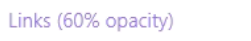
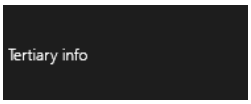
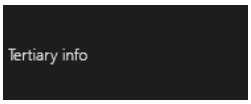
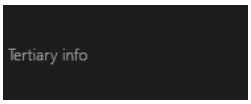
この書体見本 (type ramp) は、UI の各部分でどのサイズ、太さ、色を使うかを示しています。UI の部分ごとに、濃色スタイル シートを使う場合と淡色スタイル シートを使う場合にどの色を使うかを示しています。

- プライマリ テキストの不透明度は 100% です。
- セカンダリ テキストの不透明度は 60% です。
- ホバー状態のテキストの不透明度は 80% です。
- 押された状態の白黒テキストの不透明度は 40% です。その他の色のテキストの不透明度は 60% です。

(各エントリの "図番号" は、この表の後の図内でその番号のテキスト スタイルを示しています。)

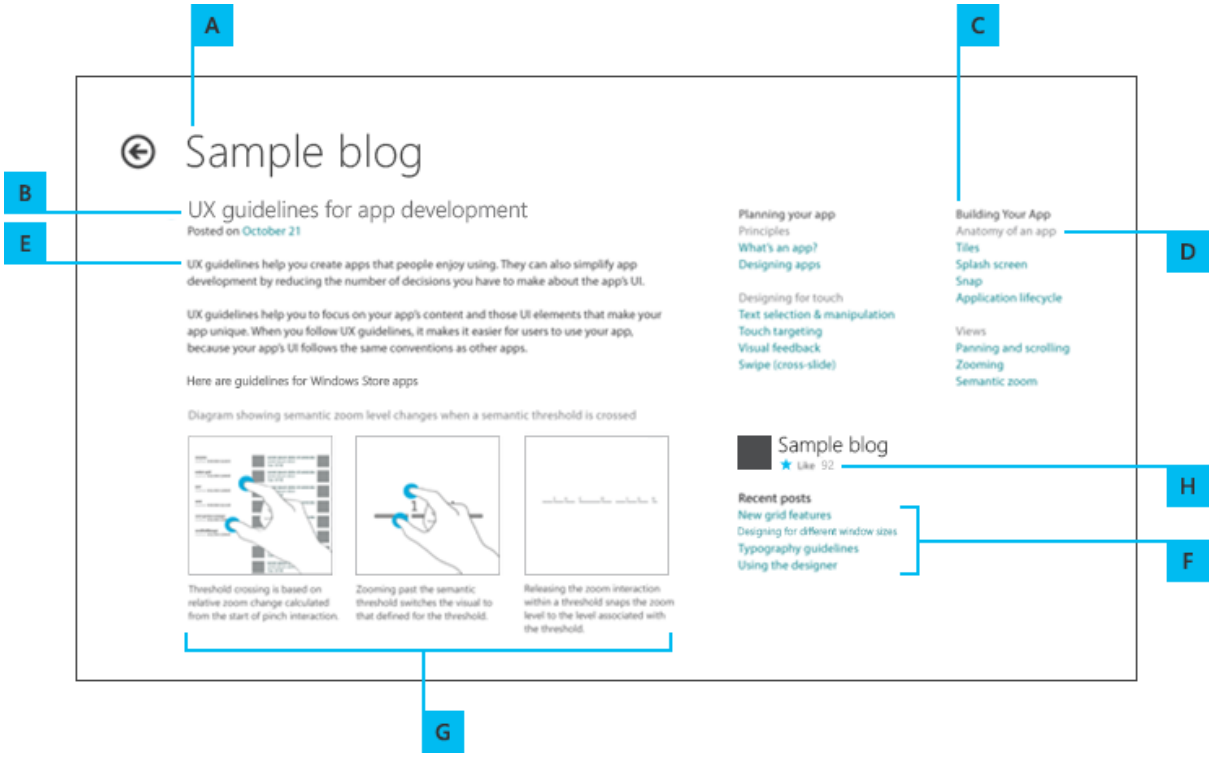
既定のビュー	通常の状態	ホバー状態 (不透明度: 80%)	押された状態 (不透明度: 40%)
ページ ヘッダー •アプリの幅が 500 ピクセル未満の場合は使わない •フォント ファミリ: Segoe UI Light •フォント サイズ: 42 ポイント •図番号: A	Header rgba(255, 255, 255, 1.0)	Header rgba(255, 255, 255, 0.8)	Header rgba(255, 255, 255, 0.4)
	Header rgba(0, 0, 0, 1.0)	Header rgba(0, 0, 0, 0.8)	Header rgba(0, 0, 0, 0.4)
ページ サブヘッダー •フォント ファミリ: Segoe UI Light •フォント サイズ: 20 ポイント •図番号: B	Subheader rgba(255, 255, 255, 1.0)	Subheader rgba(255, 255, 255, 0.8)	Subheader rgba(255, 255, 255, 0.4)
	Subheader rgba(0, 0, 0, 1.0)	Subheader rgba(0, 0, 0, 0.8)	Subheader rgba(0, 0, 0, 0.4)
対話型サブヘッダー •フォント ファミリ: Segoe UI Light •フォント サイズ: 20 ポイント •グリフ: Segoe UI Symbol U+E26B •テキストとグリフの間隔: 10 ピクセル	Subheader >* rgba(255, 255, 255, 1.0)	Subheader >* rgba(255, 255, 255, 0.8)	Subheader >* rgba(255, 255, 255, 0.4)
	Subheader >* rgba(0, 0, 0, 1.0)	Subheader >* rgba(0, 0, 0, 0.8)	Subheader >* rgba(0, 0, 0, 0.4)

項目のタイトル /ヘッダー •小さなサブヘッダー •フォント ファミリ: Segoe UI Semibold •フォント サイズ: 11 ポイント •図番号: C	 rgba(255, 255, 255, 1.0)	 rgba(255, 255, 255, 0.8)	 rgba(255, 255, 255, 0.4)
	 rgba(0, 0, 0, 1.0)	 rgba(0, 0, 0, 0.8)	 rgba(0, 0, 0, 0.4)
項目のタイトル /ヘッダー、セカンダリ •フォント ファミリ: Segoe UI Semibold •フォント サイズ: 11 ポイント •図番号: D	 rgba(255, 255, 255, 0.6)	 rgba(255, 255, 255, 0.8)	 rgba(255, 255, 255, 0.4)
	 rgba(0, 0, 0, 0.6)	 rgba(0, 0, 0, 0.8)	 rgba(0, 0, 0, 0.4)
ナビゲーション •項目のタイトル /ヘッダー、ナビゲーション •フォント ファミリ: Segoe UI •フォント サイズ: 11 ポイント	 rgba(255, 255, 255, 1.0)	 rgba(255, 255, 255, 0.8)	 rgba(255, 255, 255, 0.4)
	 rgba(0, 0, 0, 1.0)	 rgba(0, 0, 0, 0.8)	 rgba(0, 0, 0, 0.4)

本文テキスト •フォント ファミリ: Segoe UI Semilight •フォント サイズ: 11 ポイント •図番号: E	 rgba(255, 255, 255, 1.0)	なし	なし
	 rgba(0, 0, 0, 1.0)	なし	なし
リンク •フォント ファミリ: Segoe UI Semilight •フォント サイズ: 11 ポイント •図番号: F	 rgba(148, 102, 255, 1.0)	 rgba(148, 102, 255, 0.8)	 rgba(148, 102, 255, 0.6)
	 rgba(79,26,203, 1.0)	 rgba(79,26,203, 0.8)	 rgba(79,26,203 , 0.6)
三次情報 •フォント ファミリ: Segoe UI •フォント サイズ: 9 ポイント •図番号: G	 rgba(255, 255, 255, 1.0)	 rgba(255, 255, 255, 0.8)	 rgba(255, 255, 255, 0.4)
	 rgba(0, 0, 0, 1.0)	 rgba(0, 0, 0, 0.8)	 rgba(0, 0, 0, 0.4)

三次情報、セカンダリ •フォント ファミリ: Segoe UI •フォント サイズ: 9 ポイント •図番号: H	Tertiary info rgba(255, 255, 255, 0.6)	Tertiary info rgba(255, 255, 255, 0.8)	Tertiary info rgba(255, 255, 255, 0.4)
	Tertiary info rgba(0, 0, 0, 0.6)	Tertiary info rgba(0, 0, 0, 0.8)	Tertiary info rgba(0, 0, 0, 0.4)

以下の図では、Windows ストア アプリのレイアウトとタイポグラフィを基本的なブログ記事でどのように適用するかを示しています。



注 Windows の書体見本 (type ramp) は、暗色の背景に明色のテキストを表示する状態に合わせて最適化されており、あえて線が細いフォントが使われています。アプリで明色の背景に暗色のテキストを表示することが多い場合、この書体見本に従うことは適切でない可能性があります。文字サイズが小さいと、Segoe Ui Light または Semilight などのフォントは見づらくなるからです (薄い灰色の背景に濃い灰色のテキスト)。明色の背景に暗色のテ

キストを表示する場合、Regular や Semibold などの線が太いフォントの方が適しています。

推奨される書体見本に従わない場合は、書体のサイズとスタイルの種類はできる限り少なくしてください。

Segoe UI のトラッキング (文字間隔)

UI のトラッキング (全体的な文字間調整) は、特に暗く複雑な背景にテキストが表示される場合に、テキスト全体の読みやすさを左右します。

トラッキングの調整は、通常は非常に微量で、文字が非常に大きくなければほとんど気付かないほどですが、そのわずかな調整で段落やページの読みやすさが大きく変わります。

トラッキングは、ピクセルやミリのような固定単位ではなく、比例単位で指定されます。比例単位の em は、書体のサイズのポイント数と同じです。たとえば、11 ポイントの書体の em の幅は、11 ポイントです。トラッキングを設定するには、letter-spacing CSS プロパティを設定します。

Segoe UI では、フォントのサイズと太さに基づいて、次のトラッキング値をお勧めします。

サイズ	太さ	トラッキング (文字間隔) 値
42 ポイント	細い	0.00 em
20 ポイント	通常	0.01 em
その他すべて	全て	0.02 em

JavaScript 用 Windows ライブラリ スタイル シートを使う場合は、これらのトラッキング値が自動的に設定されます。

Cambria と Calibri フォントでは、既定のトラッキングを維持することをお勧めします。

Windows 8 での Segoe UI の Stylistic Set の変更点

Windows 8 では、Segoe UI の新しいバージョンが使われています。多数の既定の文字でデザインが変更され、新しい太さ、新しい Microsoft OpenType 代替文字が追加され、言語サポートが拡張されています。再デザインされた文字は、他の Windows 8 実装に、より接近して表示されますが、標準スタイルと太字スタイルでの文字の幅は変わっていません。つまり、既存のダイアログ、コンテンツ、Web サイトは、Segoe UI の新しいバージョンを使用しても、再配置する必要がありません。

次の図は、再デザインされた主な文字を示しています。

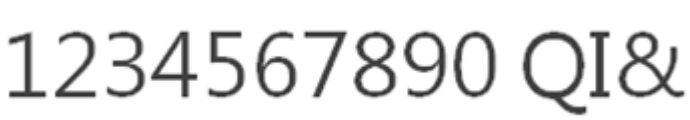
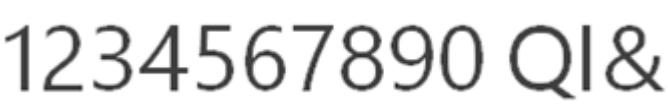
Windows 7 での Segoe UI	
Windows 8 での Segoe UI	

SS01 OpenType Stylistic Set を使うと、Windows 7 の文字を使うことができます。このセットを使うには、次の例に示すように、-ms-font-feature-settings プロパティを 'ss01' 1 に設定します。

```
<p style="font-family: 'Segoe UI'; -ms-font-feature-settings: 'ss01' 1;">
   Windows 7 Segoe UI
</p>
```

アプリで C#/VB/C++ と XAML を使う場合は、Typography.SetStylisticSet1 メソッドを使って、目的の dependency object に対する StylisticSet1 attached property の値を true に設定します。

SS20 Stylistic Set を使うこともでき、こちらのデザインと文字間隔の方をお勧めします。次の図は、既定の Windows 8 Segoe UI、SS01、SS20 のバージョンの違いを強調して示しています。

Windows 8 の既定	
SS01 (Windows 7 Segoe UI との互換性あり)	
SS20 (推奨されるデザインと文字間隔)	

下位互換性を配慮する必要がない場合は、SS20 Stylistic Set を使います。

テキスト表示用: Cambria

多くの印刷メディアではセリフ フォントが使われているため、ユーザーは書籍や雑誌を読む場合にセリフ フォントを期待します。 Cambria は、画面上での拡張された読書のために設計されたセリフ フォントです。書籍や雑誌のコンテンツのような大量のテキストを表示する場合は、Cambria を使います。

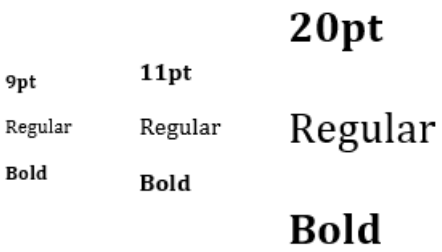
注　Cambria は書式設定されていない eBook コンテンツの既定のフォントですが、読書アプリでは読者がフォントを変えられる場合があります。また、より多くの eBook コンテンツが、デザイナーに指定された埋め込みフォントで書式設定されるようになると思われます。

読書アプリについては、次のガイドラインに従ってください。

- テキストの列の幅は、快適に読める行の長さを超えないようにします。タイポグラフィグリッドには、そのための仕様があります。
- 長いテキストは、複数の章や節に分割します。
- 読者が中断したところから再開できるようにサポートします。

Cambria フォントを使う場合は、フォントのサイズを 9 ポイント、11 ポイント、20 ポイントにします。標準の太さの文字と太字を使うことができます。既定のトラッキングを維持することをお勧めします。

Segoe UI フォントとは異なり、Cambria では、文字の太さの異なるバージョンが WinJS に用意されていません。代わりに、フォントの太さ (font-weight CSS プロパティ。XAML を使う場合は FontWeight) を normal または bold に設定します。次に、推奨される Cambria のサイズと太さの例を示します。



テキストの表示と入力用: Calibri

Calibri は、画面上で長いテキストを読むために設計されたサンセリフ フォント ファミリです。メールやチャット クライアントのテキストのように、ユーザーが書いたり編集したりするテキストには、Calibri を使います。これが、Microsoft Outlook、Microsoft Word、Microsoft PowerPoint の既定のフォントです。

Calibri を使う場合は、フォント サイズを 13、太さを標準に設定します。Calibri では、既定のトラッキングを維持することをお勧めします。

次に、Calibri フォントの例を示します。

Calibri 13pt normal

注 13 ポイントの Calibri は、11 ポイントの Segoe UI と同じ高さなので、同じ行で使うとサイズが揃います。

フォーム レイアウトのガイドライン

優れたタッチ エクスペリエンスを提供し、画面上の領域の使用を最適化し、Windows ストア アプリでパンとスクロールを最小限にするフォームを設計します。

推奨と非推奨

- コンテンツとアプリに適したフォーム レイアウトを使います。
- フォーム内のすべてのコントロールで、ラベルの配置スタイルを同じにします。
- フォームのコンテンツが単純またはわかりやすい場合は、インライン プレース ホルダーを使います。
- 垂直方向に広いパンがある場合は、複数の列を使わないでください。
- ラベルの長さがさまざまに異なる場合は、左側にラベルを配置しないでください。
- タッチ入力が行われない場合、タッチ キーボードを自動的に起動しないでください

その他の使い方のガイダンス

フォームとコントロールのレイアウトを設計するときは、ユーザーにフォームでどのように入力してもらうかと、パンとスクロールがエクスペリエンスにどのような影響を与えるかについて考慮します。 使用する場合は、タッチ キーボード (横長で、画面の最大 50% を使用できます) とインライン エラー通知の影響も考慮します。

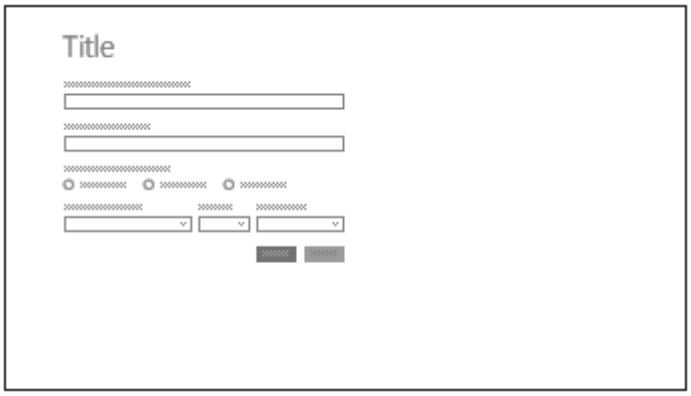
ここでは、1 列と段組みのフォーム レイアウトのガイドラインと推奨事項について説明します。

1 列のレイアウト

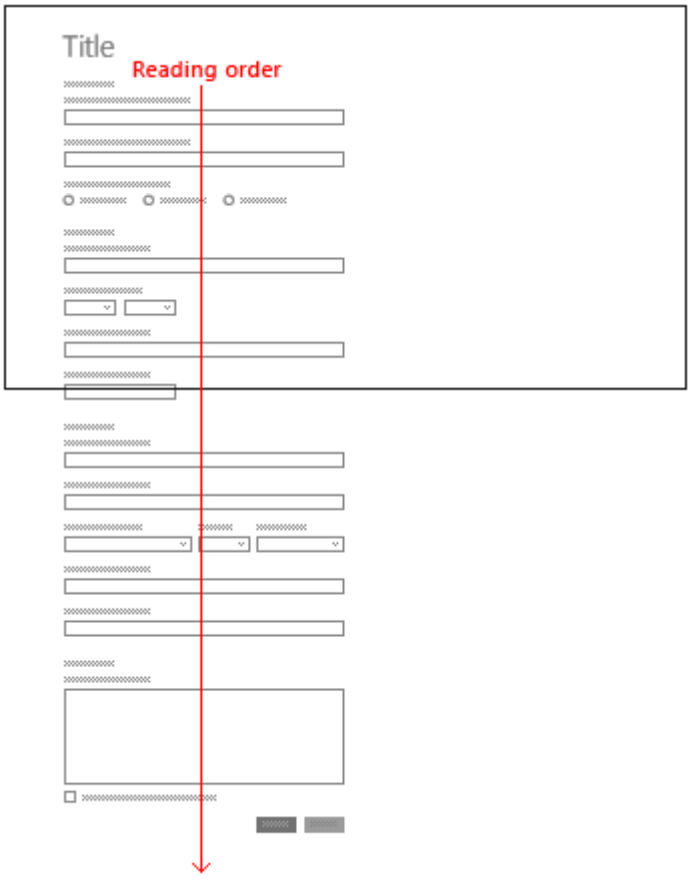
次の場合には、フォームで 1 列のレイアウトを使います。

- ユーザーに特定の順序でフォームに入力してもらいたい場合。
- フォームが複数のページにわたる場合。
- アプリは縦長で幅の狭いレイアウトに合わせてサイズ変更されます。
- アプリが追加のコメント、情報、指示、ブランド情報、広告を表示する場合。

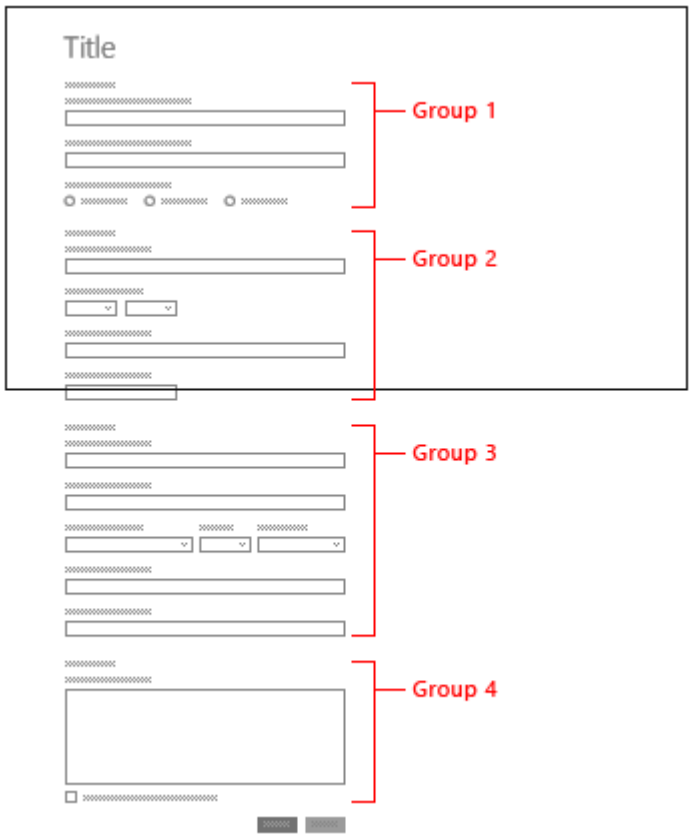
1 列のレイアウトを使った短いフォームの例を次に示します。



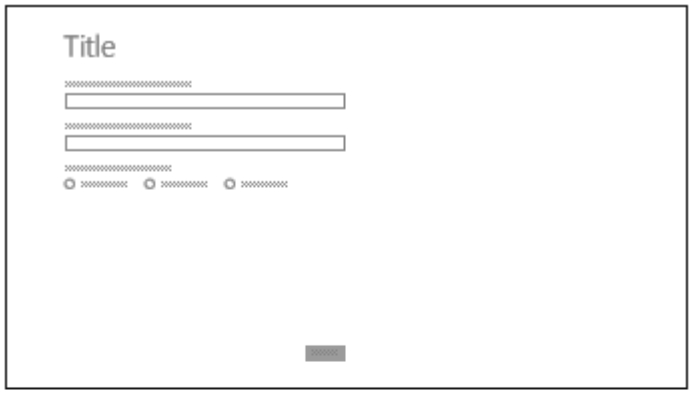
1 列のレイアウトを使った長いフォームの例を次に示します。

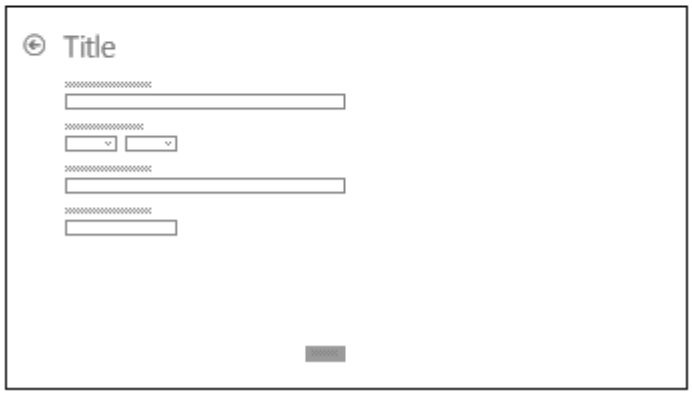


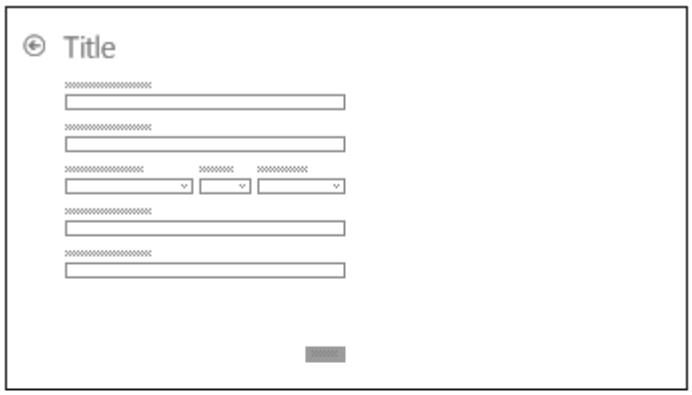
多数のコントロールを 1 つの非常に長いフォームに収めようとするのではなく、タスクをグループや一連の複数のフォームに分けることを検討してください。グループに分けた長いフォームの例を次に示します。

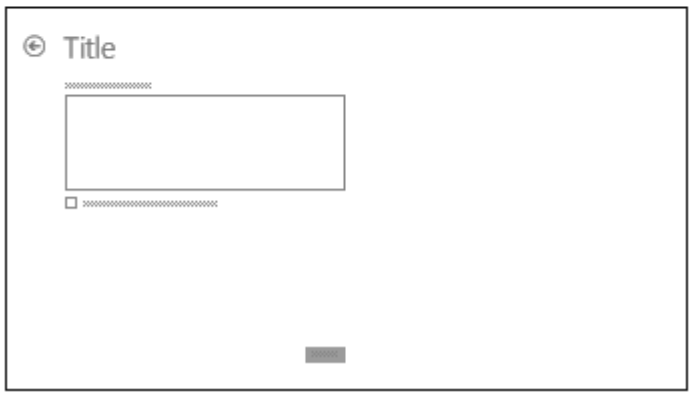


複数のページに分けた長いフォームの例を次に示します。

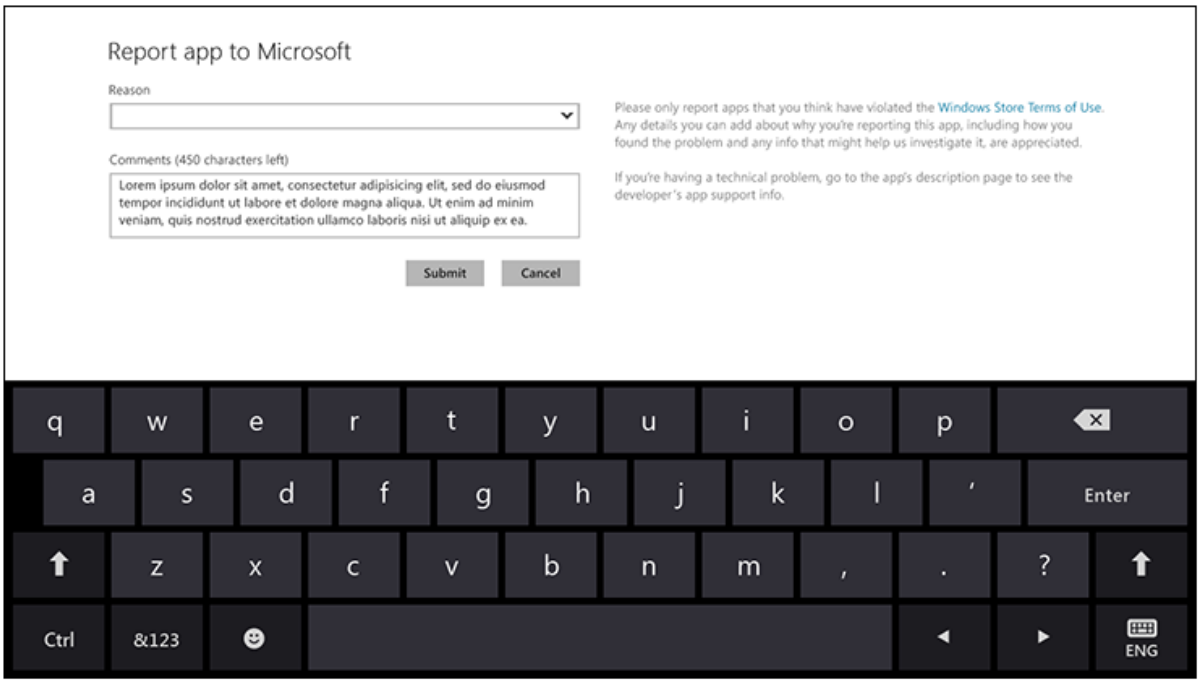








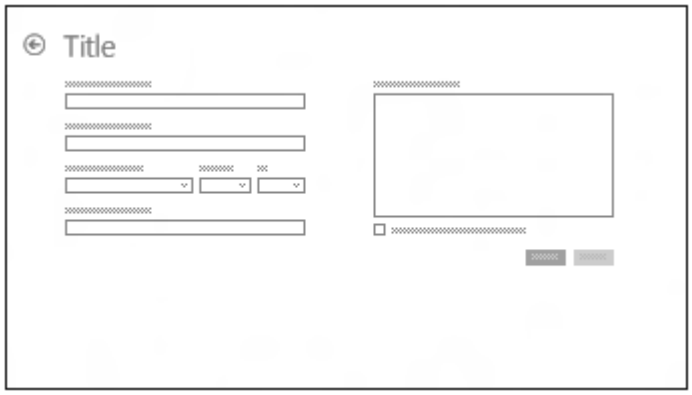
UI に追加のコメントと情報が含まれている場合に 1 列のレイアウトを使ったフォームの例を次に示します。



2 列のレイアウトを使う場合

垂直方向のパンに制限がある短いフォームの場合は、2 列のレイアウトを使います。 2 列のレイアウトでは、横方向の画面領域を最大限に活用します。 2 つの列の間には必ず十分な空間を確保してください。

2 列を使ったフォームの例を次に示します。



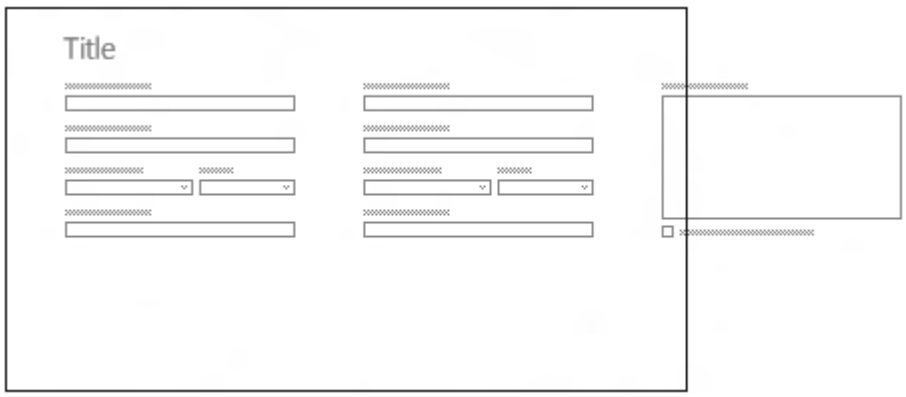
垂直方向に広いパンがある場合は、複数の列を使わないでください。フォームに入力するために、ユーザーは最初の列の一番下まで移動し、2 番目の列の先頭に戻って、また下に移動

しなければなりません。 このエクスペリエンスは、タッチ キーボードが表示されている場合、さらに扱いにくいものになります。

2 列を不適切に使ったフォームの例を次に示します。

3 列以上のレイアウトを使う場合

3 列以上のレイアウトは、横長で表示できる画面領域を最大限に活用するために使います。このレイアウトは、固定の画面または水平方向にパンする画面ではうまく機能しますが、垂直方向にパンする画面では扱いにくくなります。 入力順序が重要でない場合にのみ、このレイアウトを使ってください。



ラベルの配置

ほとんどのコントロールにはラベルが必要です。

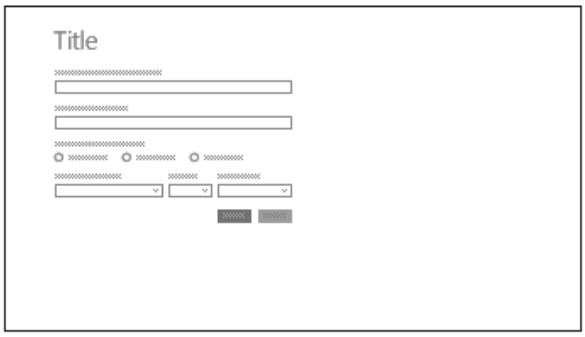
- コントロールの上側またはコントロールの左側にラベルを配置します。
- フォーム内のすべてのコントロールで、ラベルの配置スタイルを同じにします。

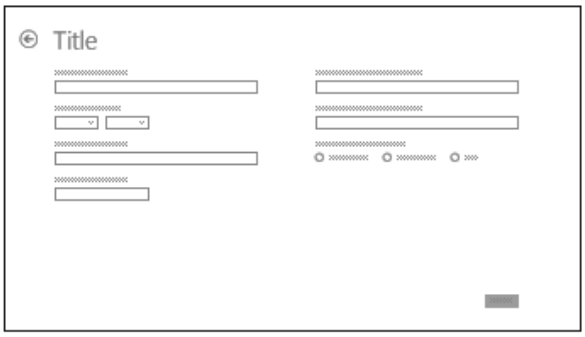
上側にラベルを配置する場合

通常は、コントロールの上側にラベルを配置します。段組フォーム レイアウトを使う場合は、常にコントロールの上側にラベルを配置します。

コントロールの上側にラベルを配置すると、ラベルとコントロールの関係を確立しやすくなり、すべてのラベルとコントロールを左揃えにできるので、すっきりしたレイアウトになります。コントロールの上側にラベルを配置すると、長い文字列を表示することや、ローカライズとアクセシビリティの問題に対処することが容易になります。

以下に、上側のラベル配置の例を 2 つ示します。



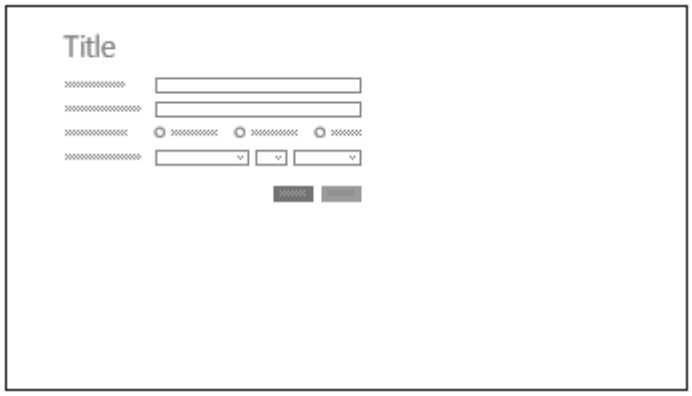


左側にラベルを配置する場合

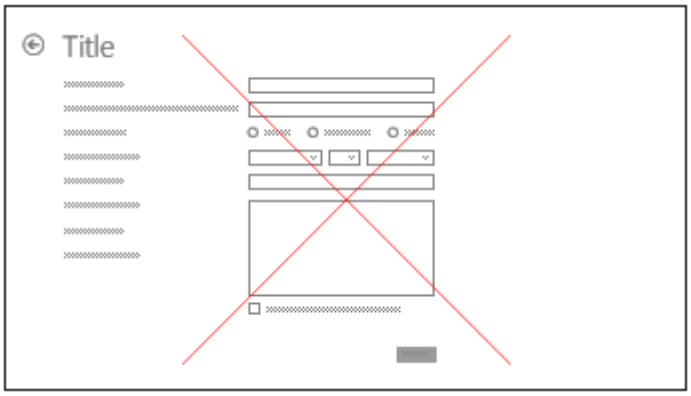
1 列のフォームで垂直方向の領域を節約する必要がある場合、次のときにはコントロールの左側にラベルを配置します。

- それぞれのラベル文字列が短く、長さがほぼ同じである。
- どのロケールでも、最も長いラベル文字列を表示するのに十分な横方向の領域がある。

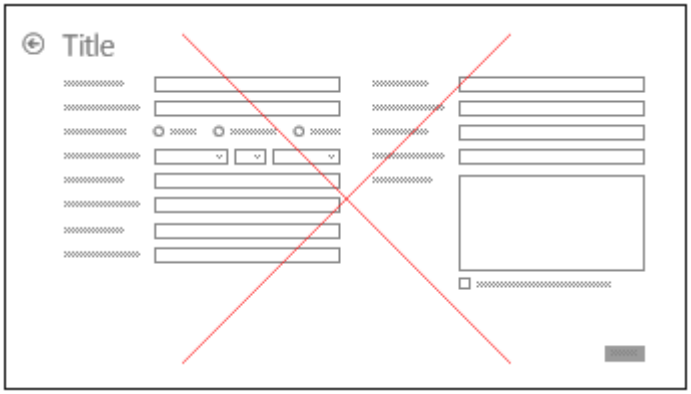
左側にラベルを配置したフォームの例を次に示します。



ラベルの長さがさまざまに異なる場合は、一部のラベルがコントロールから離れすぎた位置になるため、左側にラベルを配置しないでください。



段組フォーム レイアウトでは、左側のラベル配置を使わないでください。ラベル自体が別の列のように見える場合があります。

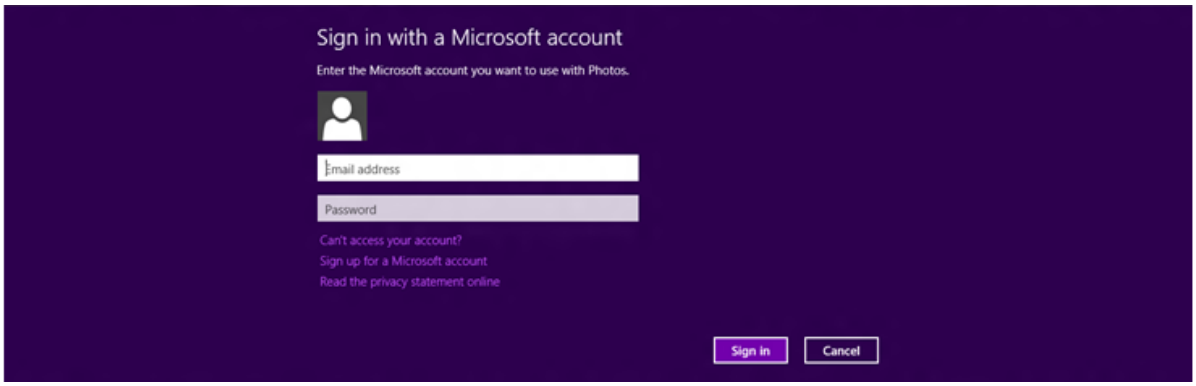


インライン プレースホルダー テキスト

ラベルの代わりにインライン プレースホルダー テキストを使うと、レイアウトが簡単になることがあります。インライン プレースホルダー テキストを使うのは次のような場合です。

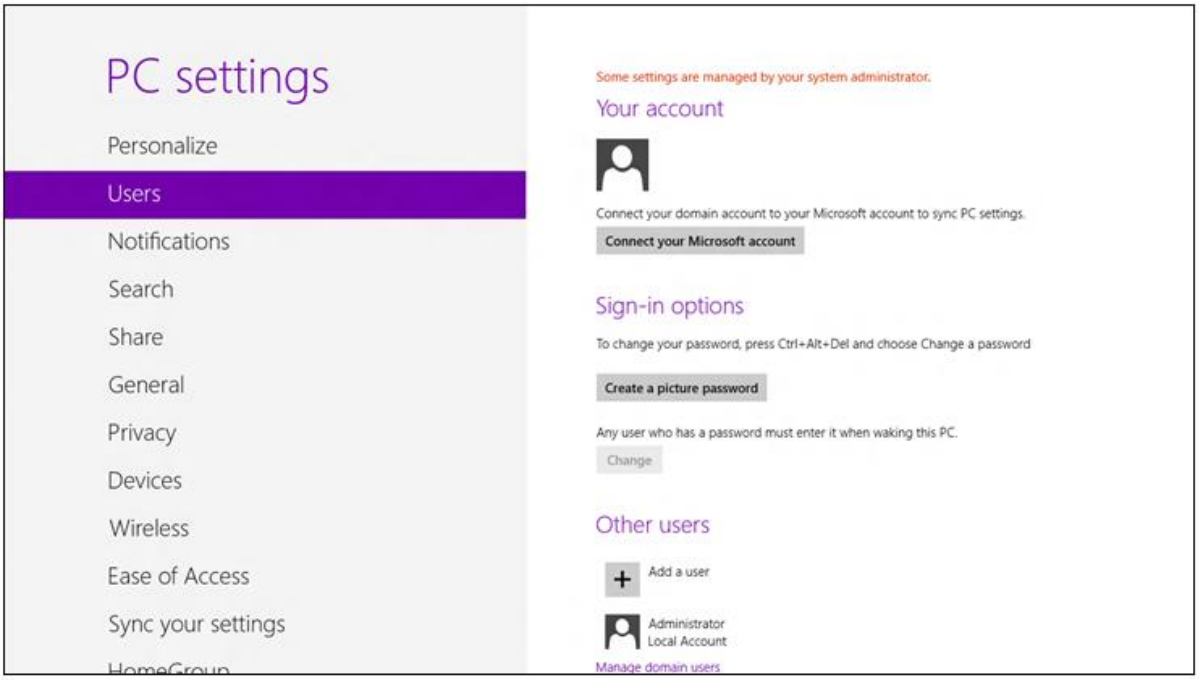
- フォームのパターンが多数のユーザーに一般的に理解されている場合 (たとえば、ユーザーのログオン制御やパスワード入力のフィールド)。
- 入力フィールドへのデータ入力後に簡単にラベルを推量したり解釈したりできる場合 (データの入力後にツールチップのテキストが非表示になるため)。
- 入力フィールドの数に制限がある場合。

インライン プレースホルダー テキストを使ったフォームの例を次に示します。



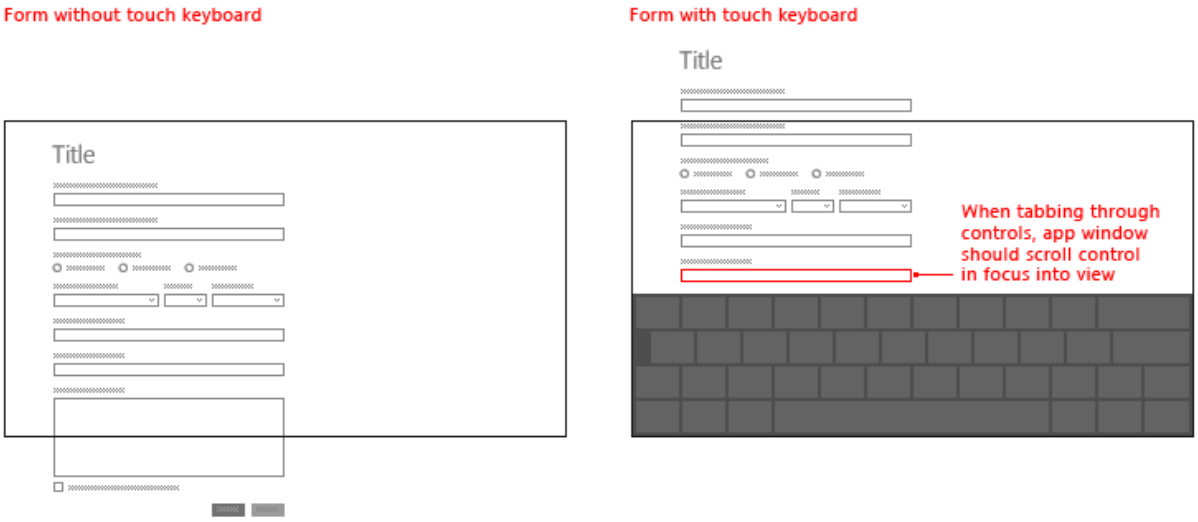
ボタンの配置

複数のボタンが他のコントロールに埋め込まれている場合以外は、フォームのボタンは右揃えにします。 フォームをより整理された外見にするには、ボタンを他のコントロールの配置と一致させる必要があります。 たとえば、[PC 設定] の画面では、埋め込まれたボタンは左揃えになっています。

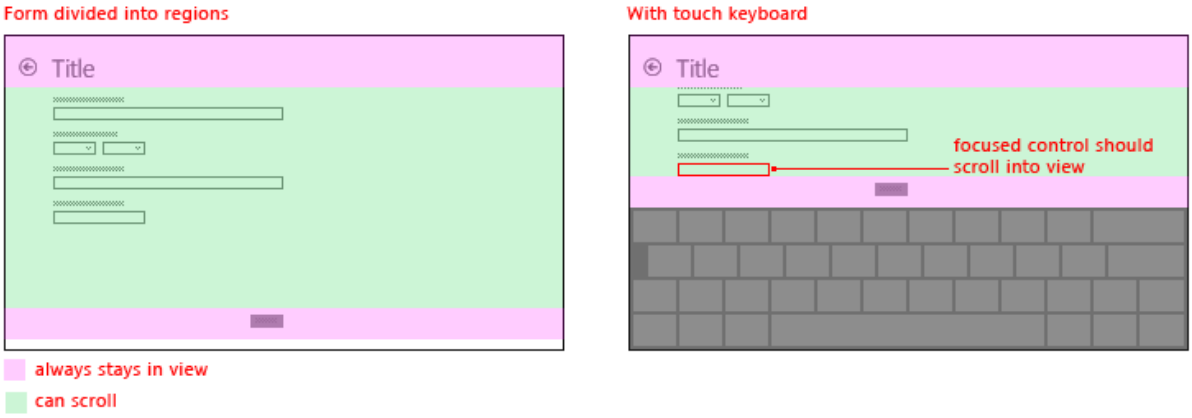


フォーカスとナビゲーション

ユーザーがフォームで操作しているコントロールにはフォーカスがあります。 ユーザーは、キーボードの Tab キーを使って、フォームのコントロール (現在は表示されていないコントロールを含む) 間を移動します。 フォーカスのあるコントロールが含まれるパン領域は、コントロール全体が表示されるように自動的にパンされる必要があります。 フォーカスがあるコントロールと、画面の端部またはタッチ キーボード (表示されている場合) の上部との間は 30 ピクセル以上ある状態にして、さまざまなエッジ ジェスチャー、UI、テキスト選択グリッパー用にスペースを残しておく必要があります。



場合によっては、画面にずっと表示されたままであることが必要な UI 要素もあります。フォーム コントロールがパン領域に含まれ、重要な UI 要素が静的であるように UI を設計します。 たとえば、次のような場合です。



タッチ キーボードの起動と終了

物理的なキーボードが検出されない場合、ユーザーがテキスト入力フィールドをタップすると、タッチ キーボードが表示されます。 この操作が行われない場合は、アプリがタッチ キーボードを自動的に起動しないようにしてください。

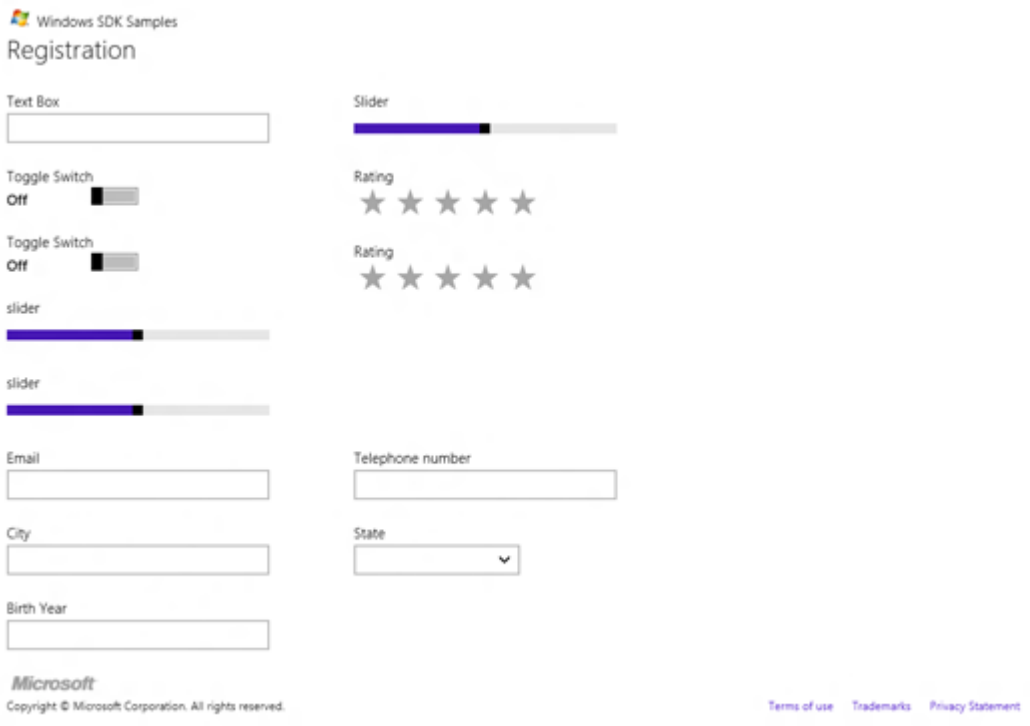
キーボードは次のいずれかの方法で終了します。

- ユーザーがフォームへの入力を完了してフォームを送信し、そのビューから移動する。
- ユーザーが終了キーを使ってキーボードを閉じる。

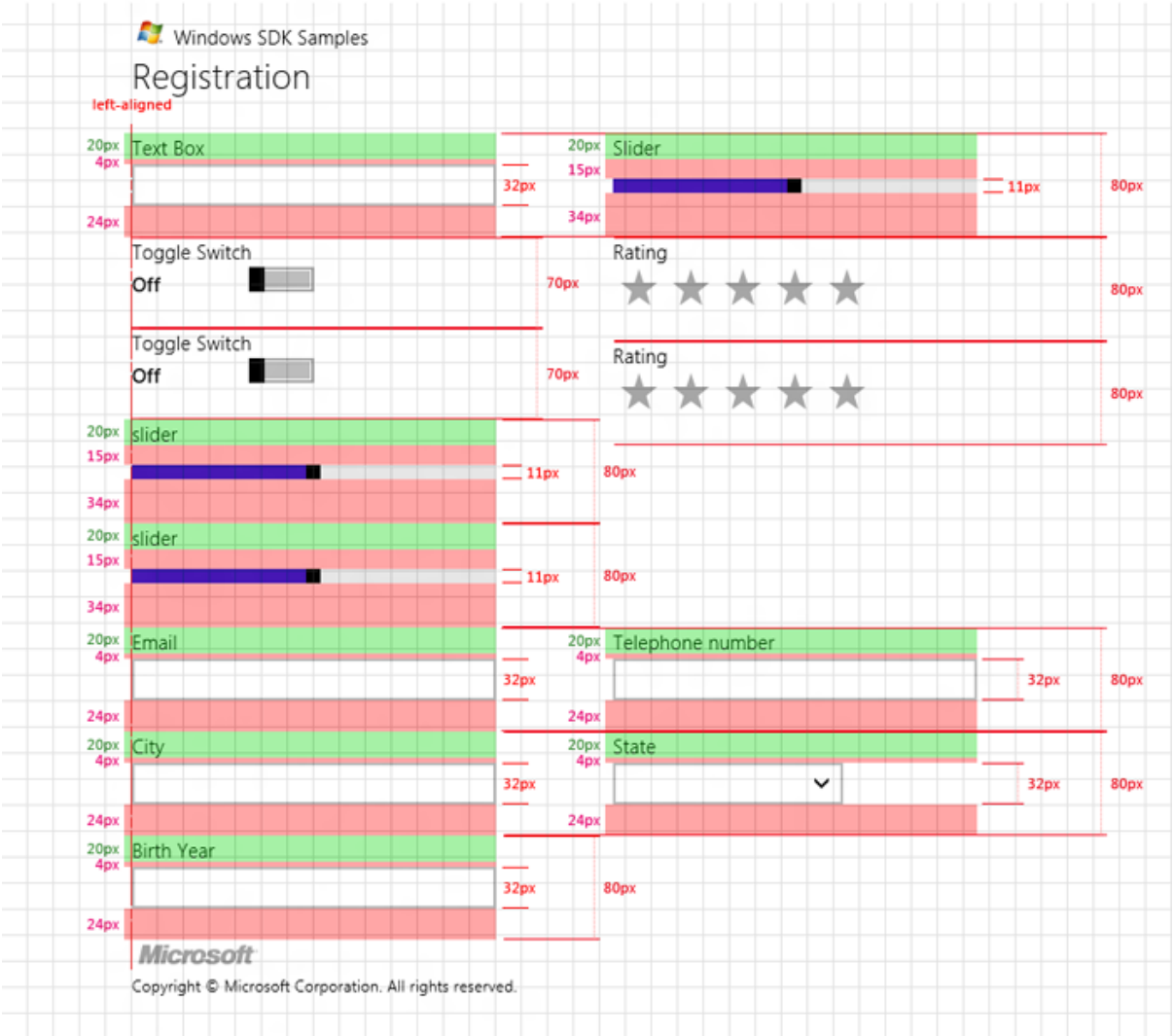
ユーザーがフォームでコントロール間を移動している間は、タッチ キーボードは自動的に終了しません。

レイアウトの例

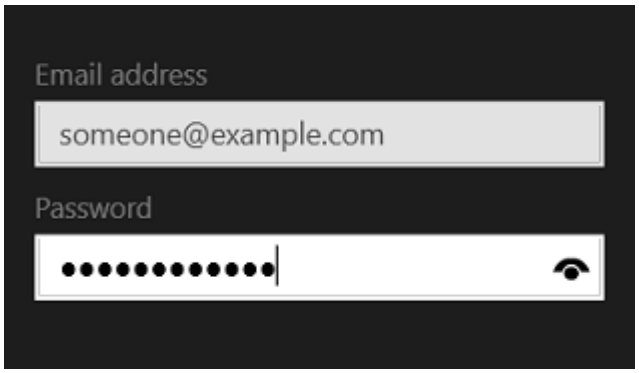
このドキュメントのガイドラインに従った登録フォームの例を次に示します。



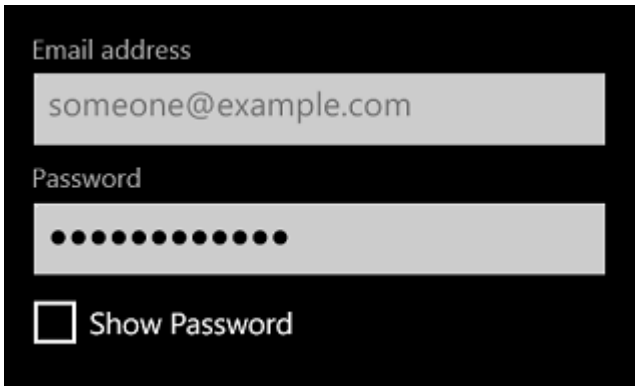
次に、さまざまな要素の推奨サイズと、間隔のガイドラインを示します。



パスワード ボックスのガイドライン



Windows ストア アプリ



Windows Phone ストア アプリ

説明

パスワード ボックスは、プライバシーの目的で入力文字が非表示になるテキスト入力ボックスです。

パスワード ボックスは、入力されたテキストの代わりに記号が表示される点を除けば、テキスト入力ボックスに似ています。この記号は、構成できます。

ユーザーがコントロールを操作するか、変更が保存された後も、記号は引き続き表示されます。ただし、引き続き表示される記号の数は、必ずしも入力文字の数には一致しません。

ユーザーが既に入力されているパスワード ボックスをタップすると、既存の記号が強調表示され、既存の値を入力で置き換えることができるようになります。

Windows Phone では、ユーザーによって入力された各文字がわずかの間表示された後、記号に正規化されます。

適切なコントロールの使用

パスワード ボックスを使うことで、ユーザーが非常に重要な機密データを入力しているときに、だれも肩越しにそれを見ることができなくなります。パスワード ボックスは、テキスト入力ボックスの高機密性のバージョンです。

パスワード ボックスでは、パスワードに加えて、社会保障番号などその他の機密データも収集できます。ただし、入力フィードバックを削減すると、入力間違いを発見することが難しくなります。2 ～ 3 語以上の単語を入力する場合は、ユーザーが入力内容を一時的に見ることができるように、パスワード表示ボタンを有効にすることを考えます。

推奨と非推奨

- ユーザーがアカウントを作成している場合で、ユーザー名とパスワードを入力するときは、パスワード入力用およびパスワード確認用の 2 つのパスワード ボックスを提示することを考慮します。
- ユーザーがログオンしている場合は、単一のパスワード ボックスのみを提示します。
- ユーザーが PIN を入力している場合は、確認ボタンを押す必要なく、最終的な正しい文字が入力された瞬間に直ちに応答することを考慮します。このために、ログオン ダイアログを閉じるか、ワークフローの次のページに移動することが考えられます。

Segoe UI Symbol アイコンのガイドライン

このドキュメントでは、Segoe UI Symbol フォントが提供する、アイコンとして使うことができる便利なグリフを示します。

推奨と非推奨

- これらのグリフは、Segoe UI Symbol フォントを指定できる場合にのみ使います。

その他の使い方のガイダンス

Segoe UI Symbol に追加された UI コントロールのほとんどは、Unicode の私用領域 (PUA) にマップされます。フォント開発者は PUA を使って、既にあるコード ポイントにマップされないグリフにプライベート Unicode 値を割り当てることができます。これは、記号フォントを作成するときに役立つ場合がありますが、相互運用性の問題が生じます。フォントが利用できない場合、グリフは表示されません。Segoe UI Symbol フォントを指定できる場合にのみ、これらのグリフを使います。

タイルを使っている場合は、タイルのフォントを指定できず、フォントのフォールバックで PUA グリフを使うことができないため、これらのグリフは使うことができません。

これらのグリフの多くは、幅が 0 で使うことができます。幅が 0 のグリフは、互いに重ね合わせることができます。たとえば、幅が 0 の、赤で塗りつぶされたハート (U+E00B) を挿入した場合、幅が 0 のため、カーソルはハートの末尾に移動しません。次に、黒い輪郭のみのハート (U+E006) を挿入すると、輪郭のみのハートが塗りつぶされたハートの上に重ねられます。

また、アイコンの多くは、双方向テキストを使う言語で利用できるミラー形式に対応します。

C#/VB/C++ と XAML を使ってアプリを開発している場合、Unicode ID の代わりに Symbol 列挙値を使って Segoe UI Symbol フォントのグリフを指定できます。JavaScript と HTML を使ってアプリを開発している場合、Unicode ID の代わりに AppBarIcon 列挙値 を使って Segoe UI Symbol フォントのグリフを指定できます。

ハート

U+E006	♡	Outlined heart
U+E0A5	♥	Solid heart
U+E007	💔	Broken heart
U+E00B	♥	Solid heart (zero width)
U+E00C	💔	Broken heart (zero width)

星評価

U+E224	☆	Outlined star
U+E0B4	★	Solid star
U+E00A	★	Solid star (zero width)
U+E082	★	Solid star (reduced padding for use in HTML)
U+E0B5	★	Small star

チェック ボックス コンポーネント

U+E001	✓	Check mark
U+E002	■	Fill
U+E003	□	Box
U+E004	■	Indeterminate state
U+E005	☑	Reversed
U+E008	✓	Check mark (zero width)
U+E009	■	Fill (zero width)
U+E0A2	☑	Composite

ラジオ ボタン コンポーネント

U+E070		Outer circle (zero width)
U+E072		Inner circle (zero width)
U+E080		Fill (zero width)
U+E0A3		Composite

その他

U+E206		Comment
U+E208		Favorite
U+E209		Like
U+E20A		Video
U+E20B		Mail message
U+E248		Reply
U+E249		Favorite
U+E24A		Unfavorite
U+E25A		Mobile contact
U+E25B		Blocked contact
U+E25C		Typing indicator
U+E25D		Video presence chicklet
U+E25E		Presense chicklet

スクロール バーの矢印

U+E00E		U+E010	
U+E00F		U+E011	
U+E016		U+E018	
U+E017		U+E019	

段階的表示の矢印

U+E096		U+E098	
U+E097		U+E099	
U+E09A		U+E09C	
U+E09B		U+E09D	
U+E012		U+E014	
U+E013		U+E015	
U+E09E		U+E0A0	
U+E09F		U+E0A1	
U+E26B		U+E26C	
U+E228			

戻るボタン

戻るボタンのグリフは 3 つの異なるサイズのものを使うことができるため、16 ポイント、20 ポイント、42 ポイントで外側の輪の太さの一貫性を確保できます。これらのグリフは重ねることができます。

16pt コード	20pt コード	42pt コード	記号	説明
U+E2A4	U+E0BA	U+E071		戻るボタン
U+E2A5	U+E0BB	U+E0A6		戻るボタンの外側の円
U+E2A6	U+E0C4	U+E0A7		戻るボタンの矢印
U+E2A7	U+E0D4	U+E0A8		戻るボタンの塗りつぶし
U+E2A8	U+E0B3	U+E08E		白黒が反転した戻るボタン
U+E2A9	U+E0AC	U+E0AA		左右が反転した戻るボタン
U+E2AA	U+E0AD	U+E0AB		左右が反転した戻るボタンの矢印
U+E2AB	U+E0AF	U+E257		左右と白黒が反転した戻るボタン

HTML の戻るボタン

これらのグリフの周囲に円を作成するには別のコードを追加します。

U+E0D5		Arrow intended for use in HTML
U+E0AE		Mirrored version of U+E0D5

AppBar のグリフ

これらのグリフをアプリ バーで使います。これらのグリフは、14 ポイントで表示するように設計されています。これらのグリフの周囲に円を作成するには別のコードを使います。

U+E100		Previous	U+E16E		Filter
U+E101		Next	U+E16F		Copy
U+E102		Play	U+E170		Emoji2
U+E103		Pause	U+E171		High priority
U+E104		Edit (mirrored at U+E1C2)	U+E172		Reply (mirrored at U+E1AF)
U+E105		Save	U+E173		Slideshow
U+E106		Clear	U+E174		Sort
U+E107		Trash	U+E179		All applications (mirrored at U+E1EC)
U+E108		Remove	U+E17A		Disconnect network drive
U+E109		Add	U+E17B		Map network drive
U+E10A		Cancel	U+E17C		Open new windows
U+E10B		Accept	U+E17D		Open with (mirrored at U+E1ED)
U+E10C		More	U+E17E		Circle for Direct UI
U+E10D		Redo	U+E17F		Fill for Direct UI
U+E10E		Undo	U+E181		Status/Presence
U+E10F		Home	U+E182		Options
U+E110		Up	U+E183		SkyDrive
U+E111		Forward	U+E184		Calendar
U+E112		Back	U+E185		Font

U+E113		Favorite	U+E186		Font color
U+E114		Camera	U+E187		Status
U+E115		Settings	U+E188		Browse by album (mirrored at U+E1C1)
U+E116		Video	U+E189		Alternate audio track
U+E117		Sync	U+E18A		Placeholder
U+E118		Download	U+E18B		View
U+E119		Mail	U+E18C		Set as lock screen image
U+E11A		Find	U+E18D		Set as tile image
U+E11B		Help (mirrored at U+E1F3)	U+E18E		Closed caption, Japan
U+E11C		Upload	U+E18F		Closed caption, Euro
U+E11D		Emoticon	U+E190		Closed caption
U+E11E		Layout	U+E191		Autoplay stop
U+E11F		Leave multiple party conversation	U+E192		Permissions
U+E120		Forward (mirrored at U+E1A8)	U+E193		Highlight
U+E121		Timer	U+E194		Disable updates
U+E122		Send/Send calendar event (mirrored at U+E1A9)	U+E195		Unfavorite
U+E123		Crop	U+E196		Unpin
U+E124		Rotate camera	U+E197		Open file location
U+E125		People	U+E198		Volume mute
U+E126		Close metadata (mirrored at U+E1BF)	U+E199		Italic
U+E127		Open metadata (mirrored at U+E1C0)	U+E19A		Underline
U+E128		Open in web	U+E19B		Bold

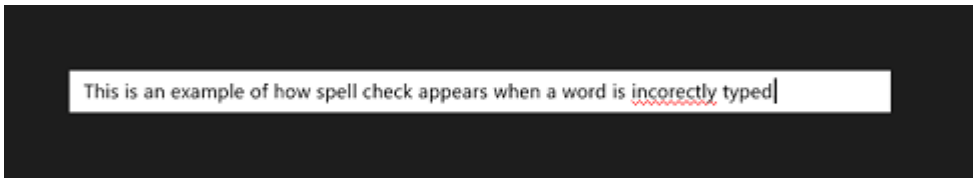
U+E129		View notifications	U+E19C		Move to folder
U+E12A		Preview link	U+E19D		Choose like or dislike
U+E12B		Category not in A-Z	U+E19E		Dislike
U+E12C		Trim video	U+E19F		Like
U+E12D		USB camera	U+E1A0		Align right
U+E12E		Zoom	U+E1A1		Align center
U+E12F		Bookmarks (mirrored at U+E1EE)	U+E1A2		Align left
U+E130		PDF	U+E1A3		Zoom neutral
U+E131		Password protected PDF	U+E1A4		Zoom out
U+E132		Page	U+E1A5		Open file
U+E133		More options (mirrored at U+E1EF)	U+E1A6		Run as other user
U+E134		Post	U+E1A7		Run as admin
U+E135		Mail2	U+E1B0	C	Italic (Italian)
U+E136		Contact Info	U+E1B1	G	Bold (Italian, French)
U+E137		Hang up	U+E1B2	S	Underline (Italian, Spanish, French, Portugese, Brazillian Portugese)
U+E138		View all albums	U+E1B3	F	Bold (Danish, German, Swedish)
U+E139		Map address	U+E1B4	K	Italic (Danish, German, Swedish, Russian, Spanish)
U+E13A		Phone	U+E1B5	U	Underline (Danish, German, Swedish)
U+E13B		Video chat	U+E1B6	I	Italic (French, Portugese, Brazil-lian Portugese)
U+E13C		Switch	U+E1B7	N	Bold (Spanish, Portugese, Brazil-lian Portugese)
U+E13D		Presence	U+E1B8	Ч	Underline (Russian)
U+E13E		Rename	U+E1B9	Ж	Bold (Russian)

U+E13F		Expand Tile (mirrored at U+E176)	U+E1BA		Font Style (Korean)
U+E140		Reduce Tile (mirrored at U+E177)	U+E1BB		Underline (Korean)
U+E141		Pin	U+E1BC		Italic (Korean)
U+E142		View all info	U+E1BD		Bold (Korean)
U+E143		Go (mirrored at U+E1AA)	U+E1BE		Font Color (Korean)
U+E144		Keyboard	U+E1C3		Street
U+E145		Dock (mirrored at U+E1AB)	U+E1C4		Map
U+E146		Charms bar (mirrored at U+E1AC	U+E1C5		Clear selection (mirrored at U+E1F4)
U+E147		App bar	U+E1C6		Font size decrease
U+E148		Remote desktop home	U+E1C7		Font size increase
U+E149		Refresh	U+E1C8		Font size
U+E14A		Rotate	U+E1C9		Cell phone
U+E14B		Shuffle	U+E1CA		Retweet (full size)
U+E14C		Details (mirrored at U+E175)	U+E1CB		Tag
U+E14D		Shop	U+E1CC		Repeat once
U+E14E		Select all	U+E1CD		Repeat/Loop
U+E14F		Rotate	U+E1CE		Favorite star empty
U+E150		Import (mirrored at U+E1AD)	U+E1CF		Add to favorite
U+E151		Import all files (mirrored at U+E1AE)	U+E1D0		Calculator
U+E153			U+E1D1		Direction
U+E154		Show all files	U+E1D2		Current location/GPS
U+E155		Browse photos	U+E1D3		Library
U+E156		Take webcam picture	U+E1D4		Address book

U+E158		Picture library	U+E1D5		Voicemail/Memo
U+E159		Save local	U+E1D6		Microphone
U+E15A		Caption	U+E1D7		Post update
U+E15B		Stop	U+E1D8		Back to window
U+E15C		Results (find) (mirrored at U+E1F1)	U+E1D9		Full screen
U+E15D		Volume	U+E1DA		Add new folder
U+E15E		Tools	U+E1DB		Calendar reply
U+E15F		Start chat	U+E1DD		Unsync folder
U+E160		Document	U+E1DE		Report hacked
U+E161		Day	U+E1DF		Sync folder
U+E162		Week	U+E1E0		Block contact
U+E163		Month (mirrored at U+E1DC)	U+E1E1		Remote desktop switch apps
U+E164		Match options	U+E1E2		Add friend/view invites
U+E165		Reply all (mirrored at U+E1F2)	U+E1E3		Remote desktop touch pointer
U+E166		Read	U+E1E4		Remote desktop go to start
U+E167		Link	U+E1E5		Remote desktop zero bars
U+E168		Accounts	U+E1E6		Remote desktop one bar
U+E169		Show bcc	U+E1E7		Remote desktop two bars
U+E16A		Bcc	U+E1E8		Remote desktop three bars
U+E16B		Cut	U+E1E9		Remote desktop four bars
U+E16C		Attach	U+E1EA	*K*	Italic (Russian)

U+E1F5		Record
U+E1F6		Lock
U+E1F7		Unlock
U+E1FD		Arrow down
U+E28F		Save as
U+E290		Decrease indent (mirrored at U+E297)
U+E291		Increase indent (mirrored at U+E298)
U+E292		Bulleted list (mirrored at U+E299)
U+E29B		Play on
U+E294		Scan
U+E295		Preview
U+E2AC		Resolution
U+E2AD		Length
U+E2AE		Layout
U+E2AF		People
U+E2B0		Type
U+E2B1		Color
U+E2B2		Size
U+E2B3		Return to window (used in projection manager)
U+E2B4		Open content in new window (used in projection manager)

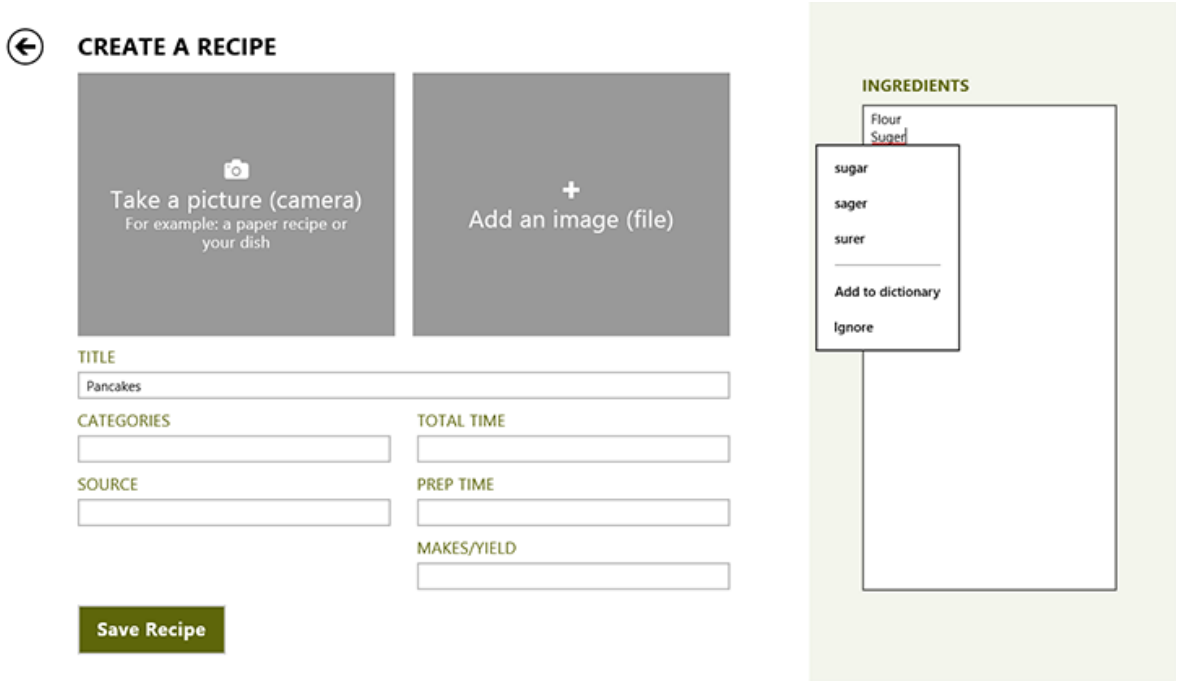
スペル チェックのガイドライン



説明

テキストの入力と編集を行っているときに、スペル チェックは単語を赤い波線で強調表示してユーザーに単語のスペルの間違いを知らせ、それを修正する方法を提供します。

例



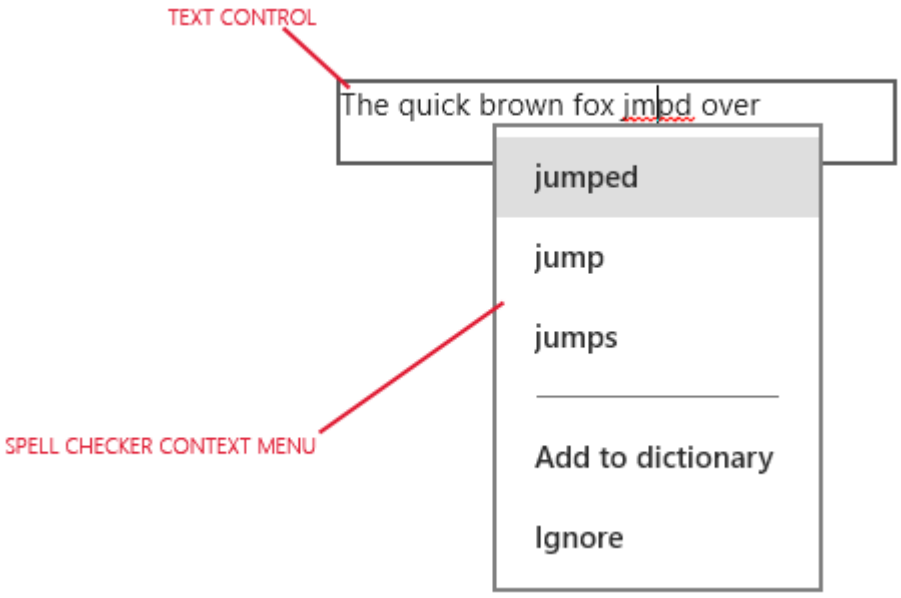
推奨と非推奨

- スペル チェックは、テキスト入力コントロールに単語や文を入力するときにユーザーを補助するために使います。スペル チェックは、タッチ、マウス、キーボード入力で機能します。

- 単語が辞書になさそうな場合や、ユーザーがスペル チェックを重視しない場合は、スペル チェックを使わないでください。たとえば、パスワード、電話番号、名前などの入力ボックスでは、スペル チェックを有効にしません。これらのコントロールでは、スペル チェックが既定で無効になっています。電話番号、パスワード、名前は、通常は辞書にないので、スペル チェックを行っても煩わしいだけです。
- 現在のスペル チェック エンジンがアプリの言語をサポートしていないという理由だけで、スペル チェックを無効にしないでください。スペル チェックでその言語がサポートされていない場合は、何も行われないので、有効にしたままで何も問題がありません。また、一部のユーザーは Input Method Editor (IME) を使ってアプリに他の言語を入力する場合があり、その言語はサポートされているかもしれません。 たとえば、中国語のアプリを構築している場合、現在はスペル チェック エンジンが中国語を認識していなくても、スペル チェックを無効にしません。ユーザーが英語 IME に切り替え、アプリに英語を入力する場合があります。スペル チェックが有効になっていれば、英語のスペル チェックが行われます。

その他の使い方のガイダンス

Windows ストア アプリでは、contentEditable プロパティが **true** に設定されている要素のために組み込みのスペル チェックが用意されています。 組み込みスペル チェックの例を次に示します。



詳しくは、JavaScript については input type=text と textarea、Extensible Application Markup Language (XAML) については TextBox class をご覧ください。

テキスト入力コントロールでのスペル チェックは、次の 2 つの目的で使います。

- スペル ミスを自動修正する
  スペル チェック エンジンは、スペルを間違えている単語を、修正が確実であれば自動的に修正します。たとえば、エンジンは自動的に "teh" を "the" に変更します。
- 代わりのスペルを示す
  修正が確実でないとスペル チェック エンジンが判断した場合、スペル ミスのある単語には赤い下線が引かれ、ユーザーがその単語をタップするか右クリックすると、コンテキスト メニューに修正候補が表示されます。

JavaScript コントロールの場合、複数行テキスト入力コントロールでは、既定でスペル チェックが有効になっており、単一行コントロールでは無効になっています。単一行コントロールでスペル チェックを手動で有効にするには、コントロールの **spellcheck** プロパティを **true** に設定します。 コントロールのスペル チェックを無効にするには、**spellcheck** プロパティを **false** に設定します。

XAML TextBox コントロールの場合、スペル チェックが既定で無効になっています。**IsSpellCheckEnabled** プロパティを **true** に設定することによって有効にできます。

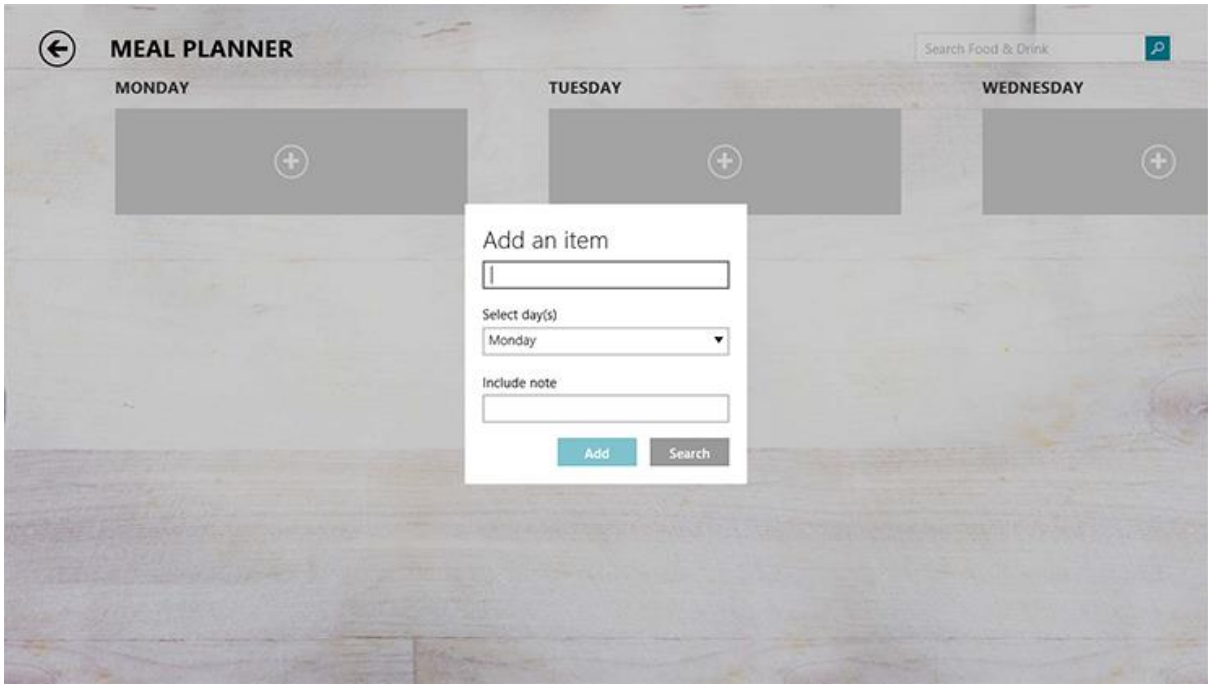
テキスト入力のガイドライン

説明

テキスト入力ボックスでは、ユーザーは物理的なキーボードやスクリーン キーボードを使ってテキストや数値を入力および編集できます。テキストの折り返しを使うことで、テキスト入力ボックスに単一行または複数行のテキストを入力できるように構成できます。

テキスト入力ボックスは、四角形の境界線とその内側の編集可能なテキスト領域で構成されます。タッチ対応デバイスでは、テキスト入力ボックスがフォーカスを受け取ると、スクリーン キーボードが表示されます。

例



適切なコントロールの選択

テキスト入力コントロールを使うと、ユーザーはテキストまたは数値を入力したり編集したりすることができます。テキスト入力コントロールを使うかどうかを判断する際には、次の点を考慮してください。

- **有効なすべての値を効率的に列挙することが現実的か?** そうである場合は、代わりにいずれかの選択コントロールを使うことを検討してください。考えられる選択コントロールは、チェック ボックス、ドロップダウン リスト、リスト ボックス、ラジオ ボタン、スライダー、トグル スイッチ、DatePicker、または TimePicker です。
- **有効な値は比較的少数か?** 特に値の文字数が多い場合は、ドロップダウン リストまたはリスト ボックスをお勧めします。
- **有効なデータに、何も制約がないか? または、形式の制約 (長さや文字の種類による制約) だけがあるか?** そうである場合は、テキスト入力コントロールを使います。このコントロールでは、入力できる文字数を制限できるほか、入力値の種類の一覧から選んで、入力できる値を特定の文字セットや文字形式 (数値、URL、通貨など) のみに制限できます。
- **値は専用のコモン コントロールがあるデータ型か?** そうである場合は、テキスト入力コントロールではなく、適切なコントロールを使います。たとえば、データ入力を受け付けるには、テキスト入力コントロールの代わりに DatePicker を使います。
- データが数値の場合:
  - **ユーザーは設定を相対的な量として認識しているか?** そうである場合は、スライダーを使います。
  - **設定の変更による影響をすぐに確認できることがユーザーにとって便利か?** そうである場合は、スライダーを使い、必要であれば付随するコントロールも使います。
  - **入力した結果の確認後 (たとえば、音量や明るさを設定した後など)、入力された値を調整する可能性が高いか?** そうである場合は、スライダーを使います。

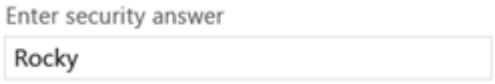
推奨と非推奨

- ユーザーがテキスト入力ボックスに入力する内容が明確でない場合、または入力に制約がある場合は、ラベルまたはプレースホルダー テキストを設定します。ラベルは、テキスト入力ボックスに値が存在するかどうかに関係なく表示します。プレースホルダー テキストはテキスト入力ボックスの内側に表示され、値を入力すると非表示になります。
- コントロールの幅が、入力できる値の範囲に適した幅になるように考慮します。言語によって単語の長さが違うことに注意し、ローカライズを考慮に入れて幅を調整します。
- テキスト入力ボックスは、通常、単一行です (テキストの折り返しがオフ)。ユーザーが長い文字列を入力または編集する必要がある場合は、テキスト入力ボックスを複数行 (テキストの折り返しがオン) に設定します。
- 通常、テキスト入力ボックスは編集可能なテキストに対して使います。ただし、読み取り、選択、コピーはできるが編集はできないように、テキスト入力ボックスを読み取り専用に設定することもできます。
- 画面をすっきりと見せる必要がある場合は、制御するチェック ボックスがオンの場合のみ、一連のテキスト入力ボックスを表示することをお勧めします。また、有効な状態のテキスト入力ボックスをチェック ボックスなどのコントロールにバインドすることもできます。
- 既に値が入力されているテキスト入力ボックスをユーザーがタップしたときの動作を検討します。既定の動作では、単語の間に挿入ポイントが配置され、何も選択されません。この動作は値の置き換えよりも編集に適しています。対象のテキスト入力ボックスが、編集よりも主に置き換えに使われる場合は、フォーカスを受け取ったときにフィールドのすべてのテキストを選択し、入力によって選択内容が置き換えられるように設定できます。
- 入力できる値を特定の文字セットまたは形式 (数値、URI、通貨など) に制限するには、入力値の種類を適切な値に設定します。これにより、使われるスクリーン キーボードが決まります。

単一行入力ボックス

- 少量のテキスト情報を多数取得する場合は、いくつかの単一行テキスト ボックスを使います。 それらのテキスト ボックスに本来の関連性がある場合は、グループ化します。
- 単一行テキスト ボックスのサイズを、予測される最長の入力より少し広くします。それによってコントロールが広くなり過ぎる場合は、2 つのコントロールに分割します。たとえば、1 つの住所の入力を "Address line 1" と "Address line 2" に分割できます。
- 最大長を設定します。バッキング データ ソースが長い入力文字列を許可しない場合は、入力を制限し、制限に達したら検証ポップアップを使ってユーザーに知らせます。
- ユーザーから少量のテキストを収集するには、単一行テキスト入力コントロールを使います。

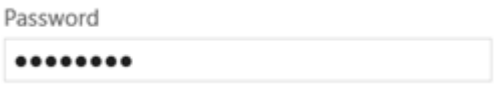
  次の例は、セキュリティの質問への回答を取得する単一行テキスト ボックスを示しています。 回答には短い文字列が想定されるので、単一行テキスト ボックスが適しています。 収集される情報は、Windows で認識される専用の入力型のいずれとも一致しないので、汎用の "Text" 型が適しています。



- 特定の書式のデータを入力するには、短い固定サイズの単一行テキスト入力コントロールのセットを使います。

- 文字列を入力または編集するには、単一行の、制約のないテキスト入力コントロールと、ユーザーが有効な値を選択できるように補助するコマンド ボタンを組み合わせて使います。

- 数値を入力または編集する場合は、単一行の数値入力コントロールを使います。
- パスワードと PIN を安全に入力するには、単一行のパスワード入力コントロールを使います。



- メール アドレスを入力する場合は、単一行のメール入力コントロールを使います。

メール入力コントロールを使う場合は、次の機能を無料で利用できます。

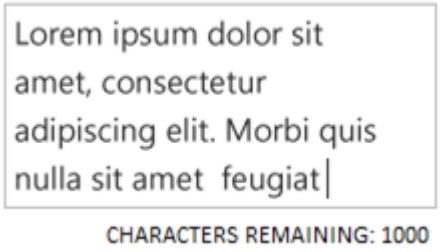
◦ ユーザーがテキスト ボックスに移動すると、タッチ キーボードがメール専用のキー レイアウトで表示されます。

◦ ユーザーが無効なメール形式を入力すると、それを知らせるダイアログが表示されます。

- Web アドレスを入力するには、URL 入力コントロールを使います。
- 電話番号を入力するには、電話番号入力コントロールを使います。
- 単一行テキスト ボックスを作成するために、行の高さを 1 にしたテキスト領域は使いません。代わりに、テキスト ボックスを使います。
- プレースホルダー テキストは、テキスト コントロールを事前設定するためには使いません。ユーザーがコントロールを使うと、プレースホルダー テキストはクリアされます。代わりに、"value" 属性を使います。
- テキスト ボックスは、検索ボックスとしては使いません。Web ページでは、検索ボックスを作成するために入力要素を使うことが一般的な習慣になっています。しかし、代わりに検索チャームを使うと、より便利で一貫性のあるエクスペリエンスを実現できます。 検索チャームは、アプリがプラグインとして使用できる、一貫性のある検索エクスペリエンスを提供します。
- パスワード入力ボックスのすぐ横に、他のコントロールを配置しないようにします。パスワード入力ボックスには、入力したパスワードをユーザーが確認するための、パスワード表示ボタンがあります。他のコントロールをすぐ横に配置すると、ユーザーが他のコントロールを操作しようとしたときに、誤ってパスワードを表示してしまう可能性があります。 これを防ぐには、パスワード入力ボックスと他のコントロールの間には少し間隔をおくか、他のコントロールを次の行に配置します。

複数行テキスト入力コントロール

- リッチ テキスト ボックスを作る場合は、スタイル設定ボタンを用意し、その動作を実装します (JavaScript を使った Windows ストア アプリでは、これらのコントロールは自動的には提供されません)。
- アプリの雰囲気に合ったフォントを使います。
- テキスト コントロールの高さは、典型的な入力が十分に入るように設定します。
- 長いテキストを取得する場合、ユーザー入力のワード数または文字数が最大値未満になるようにするには、プレーン テキスト ボックスを使い、ライブ実行のカウンターを用意して、制限までの残りの文字数または単語数をユーザーに示します。カウンターは自分で作成する必要があります。テキスト ボックスのそばに配置し、ユーザーが文字や単語を入力するたびに動的に更新します。



- ユーザーの入力中にテキスト入力コントロールの高さが増加するようにはしません。
- ユーザーが 1 行しか必要としていない場合は、複数行テキスト ボックスを使いません。
- プレーン テキストで十分な場合は、リッチ テキスト コントロールは使いません。

その他の使い方のガイダンス

テキスト入力ボックスを使うと、ユーザーがテキスト値を入力し、既に入力した値を編集することができます。また必要に応じて、入力できる値を制限できます。入力できる文字数を制限できるほか、入力値の種類の一覧から選んで、入力できる値を特定の文字セットや形式 (数値、URI、通貨など) のみに制限できます。

適切な複数行テキスト入力コントロールの選択

ユーザーが長い文字列を入力または編集する必要がある場合は、複数行テキスト コントロールを使います。複数行テキスト入力コントロールには、プレーン テキスト入力コントロールとリッチ テキスト コントロールの 2 つの種類があります。

- 複数行テキスト ボックスの主な目的がドキュメントの作成 (ブログのエントリ、メール メッセージのコンテンツなど) であり、ドキュメントでリッチ テキストが必要な場合は、リッチ テキスト ボックスを使います。
- ユーザーがテキストを書式設定できるようにする場合は、リッチ テキスト ボックスを使います。
- 内部的に使うだけで、後でユーザーに再表示しないテキストを取得する場合は、プレーン テキスト入力コントロールを使います。たとえば、アンケートを行うとします。ユーザーが回答を終了すると、データはサーバーに送信されますが、ユーザーが再度データを見ることはありません。 通常は、ユーザーがテキストのスタイルを設定できるようにする必要はありません。
- 他のすべてのシナリオでは、プレーン テキスト入力コントロールを使います。

タイポグラフィのガイドライン

説明

タイポグラフィは、Microsoft デザイン言語の中心的要素です。Microsoft デザインのすべての原則で、タイポグラフィの重要性が認識されています。アプリの開発者は、高度なタイポグラフィ機能をサポートする一連のフレームワークを初めて利用できるようになりました。

推奨と非推奨

- タイポグラフィ グリッドを使ってテキストをレイアウトします。
- Unicode を使ってテキストを改善します。
- Microsoft OpenType 機能をグローバルに適用する。
- 文の先頭文字を大文字にします。
- 数値範囲ではハイフンではなく半角ダッシュを使います。
- 推奨されるフォントを使わない場合は、随意の合字を適用しないでください。
- 両端揃えのテキストを使わないでください。

その他の使い方のガイダンス

Segoe UI

Windows の代表的なユーザー インターフェイス フォントである Segoe UI は、Windows Vista/Office 2007 の時期に導入されました。 これは、看板や標識で使われた伝統的な書体から作成されたサン セリフ デザインです。Segoe UI の導入以降、すべてのマイクロソフト製品の個性が Segoe UI によって決定付けられ、オペレーティング システム、アプリ、デバイスの印象だけでなく、ロゴ、ブランド、広告、パッケージの印象にも影響を及ぼしています。この書体は、マイクロソフトのタイポグラフィに対する方針を明確に表しています。

Windows 8 では、Segoe UI ファミリに重要なアップグレードが加えられました。主な変更点は次のとおりです。

- Segoe UI Light と Segoe UI Semibold が再ヒンティング (調整) されて、画面上でのレンダリング品質が改善されました。
- Windows Phone 版の Segoe に合わせて、既定の数字と "Q" および "I" のデザインを調整しました。SS01 Microsoft OpenType Stylistic Set を使うと、以前の Windows Vista 形式の文字を使うことができます。
- Segoe UI Light、Semilight、Semibold では、数字の幅をプロポーショナルにしました。従来の固定ピッチの数字を利用するには、C#/VB/C++ と XAML を使った Windows ストア アプリでは、Typography.NumeralAlignment 添付プロパティを "Tabular" に設定します。JavaScript と HTML を使う場合は、font-feature-settings property プロパティを "TNUM" に設定して、表形式用の数字間隔の OpenType 機能を適用します。
- Segoe UI Light は、設計時の想定よりも小さい 20 ポイント未満のサイズで使われることが多いため、線の細い Segoe UI である Segoe UI Semilight が追加されました。これは、小さいサイズでも美しくレンダリングされ、11 ポイントまで対応します。小さいフォントのテキストに軽い印象を与える場合に、このフォントをお勧めします。
- Windows Vista と Windows 7 では、Segoe UI はラテン文字、ギリシャ文字、キリル文字、アラビア文字をサポートしていました (標準および太字のみ)。Windows 8 では Segoe UI が拡張され、ヘブライ文字、アルメニア文字、グルジア文字にも対応し、アラビア文字を更新しました。また新たに、主要なヨーロッパ言語と中東言語のすべてに対応する文字セットが含められました。
- Windows UI では斜体を使いませんが、5 種類の Segoe UI すべてに斜体を追加し、アプリで使えるようにしました。
- フォントに高度な OpenType 書体 (すべての太さの小型英大文字、合字、数字のスタイルを含む) を追加しました。

他の言語のフォント

Windows 8 には、他のほとんどの言語向けの Segoe UI 調整フォントも用意されています。現在、標準と太字の 2 種類の太さのフォントを使うことができます。日本語向けの Segoe UI 調整フォントは用意されていませんが、日本語 UI フォントである Meiryo UI に

は、Segoe UI ベースのラテン文字として OpenType Stylistic Set (ss20) が含まれています。

次に、特定の言語でお勧めするフォントの一覧を示します。

文字	フォント
ラテン文字、ギリシャ文字、キリル文字	Segoe UI
アラビア語	Segoe UI
ヘブライ語	Segoe UI
アルメニア語およびグルジア語	Segoe UI
簡体字中国語	Microsoft YaHei UI
繁体字中国語	Microsoft JhengHei UI
韓国語	Malgun Gothic
タイ語	Leelawadee UI
クメール語	Leelawadee UI
ラオス語	Leelawadee UI
アメリカ先住民言語	Gadugi
アフリカ諸語	Ebrima
インド諸語	Nirmala UI
日本語	Meiryo UI

Windows 8.1 には次のフォントも用意されていることに注意してください。

- Microsoft YaHei UI Light。タイトルは 42 ポイント、見出し 2 は 20 ポイント
- Microsoft JhengHei UI。タイトルは 42 ポイント、見出し 2 は 20 ポイント
- Leelawadee UI Semilight
- Nirmala UI Semilight

アプリをローカライズするときに組み込みのフォントを使う場合は、Windows.Globalization.Fonts API に関するトピックをご覧になり、その文字システムの正しい UI とドキュメント フォントを特定してください。

代替フォントの使用

Segoe UI が Windows 8 と Microsoft を連想させることには、メリットとデメリットの両方があります。適切に使えば、アプリの信頼感を手軽に高めることができます。その一方で、このフォントを使うとアプリに独自の個性やブランド性を与えることが難しくなる場合があります。

Windows 8 には Segoe UI の代わりとなる高品質フォントが多数用意されています。Cambria などのセリフ フォントは対応言語も多く、Segoe UI と比較すると伝統的な印象を与えます。マイクロソフトの既定のオーサリング フォントである Calibri は、ユーザーが見慣れたものでしょう。 さらに、Windows 8 には Calibri に light weight が追加され、Segoe UI で sans-serif の良い代替になります。 Segoe UI と同様、 Cambria フォントと Calibri フォントには高度な OpenType 機能が備えられています。また Windows 8.1 では、一連のサイズで使用可能な Sitka と呼ばれる新しい書体ファミリが追加されました。

組み込みのフォントを使うときには、そのフォントを Windows 8 で使用できることと、Microsoft Office などの他のアプリをインストールする必要がないことを確認するために、Windows 8 をクリーン インストールしたコンピューターを使ってフォントをテストしてください。Windows 8 では、言語やエディションに関係なく、まったく同じ種類のフォントがインストールされます。

Windows の基本フォントではアプリの目的に対応できない場合には、アプリに代替フォントを埋め込むことができます。ただし、表示とライセンスに関して次のような問題が発生する可能性があることに注意してください。

- ほとんどのサード パーティ製フォントには多くの Windows 組み込みフォントと同等のヒンティングがないため、フォントを目的のサイズで徹底的にテストするようにしてください。
- ほとんどのフォント ライセンスでは、一部の形式によるドキュメント フォントの埋め込みを許可していますが、フォントを再頒布することや、アプリやゲームの内部にフォントを埋め込むことは禁止しています。フォントを埋め込む前にフォントのライセンスを熟読し、フォントの埋め込みの可否に関して疑問がある場合にはフォントの作成元に問い合わせてください。

適切なサイズと太さの選択

デスクトップ パブリッシングの出現と共に、過剰な種類の書体サイズと書体太さを簡単に扱えるようになりました。初期のデスクトップ パブリッシングのユーザーは、さまざまな書体サイズと書体太さを同じドキュメント内で使っていました。これはタイポグラフィに不慣れなユーザーの特徴でもあります。書体サイズと書体太さの種類が多すぎると、効果的な情報階層を確立することは困難です。このような理由から、Windows 8 では限られた種類のフォント サイズと太さのみを使っています。

- 見出し: Segoe UI Light 42 ポイント
- 見出し 2: Segoe UI Light 20 ポイント
- 本文: Segoe UI Semilight 11 ポイント
- キャプション: Segoe UI Regular 9 ポイント (画面解像度が一定の水準に達している場合は本文も)

すべての書体見本 (type ramp) については、「フォントのガイドライン」をご覧ください。

注 Windows の書体見本 (type ramp) は、暗色の背景に明色のテキストを表示する状態に合わせて最適化されており、あえて線が細いフォントが使われています。アプリで明色の背景に暗色のテキストを表示することが多い場合、この書体見本に従うことは適切でない可能性があります。文字サイズが小さいと、Segoe Ui Light または Semilight などのフォントは見づらくなるからです (薄い灰色の背景に濃い灰色のテキスト)。明色の背景に暗色のテキストを表示する場合、Regular や Semibold などの線が太いフォントの方が適しています。

Windows 8 の書体見本に従わない場合は、書体のサイズとスタイルの種類はできる限り少なくしてください。

アプリのページのレイアウト

テキストを囲む空間の広さはフォント サイズと同じくらい重要です。そのため、タイポグラフィグリッドを使ってテキストをレイアウトすることをお勧めします。Windows グリッ

ドの使い方について詳しくは、「アプリ ページのレイアウト」をご覧ください。 グリッドを使うだけでなく、写真やイラストの周囲の余白や空間にも特に注意してください。美しいレイアウトも、テキストの画像の回り込み処理に問題があったり、ページ要素内の余白が一貫していなかったりすると、台無しになります。

優れたタイポグラフィを実現するための 8 個のヒント

適切なフォントを適切なサイズと適切な余白で使うこと以外にも、優れたタイポグラフィを実現するためのこつがあります。タイポグラフィについては数多くの参考図書がありますが、ここではタイポグラフィを次のレベルに高めるために役立つヒントをいくつか紹介します。

1. OpenType 機能をグローバルに適用する

Segoe UI を使う場合、OpenType 機能のカーニング (kern)、随意の合字 (dlig)、Stylistic Set 20 (ss20) をアプリ内のすべてのテキストに適用します。

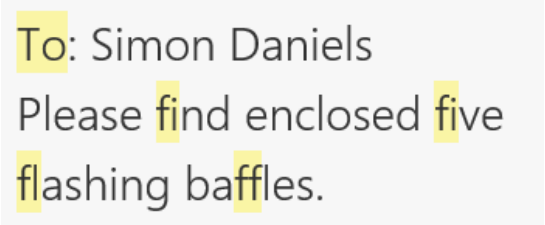
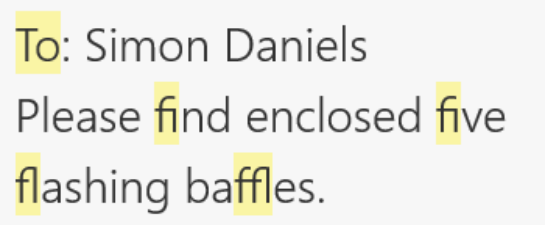
- カーニングによって、テキストの文字間隔が改善されます。これはサイズが大きい場合に顕著となります (次の図の「適用前」と「適用後」で "To" を見比べてください)。
- 合字は高品質のタイポグラフィに欠かせない要素ですが、従来の設定ではダイアログの再配置を避けるために、標準合字を Segoe UI の随意の合字としてエンコードしていました。そのため、これを使うには有効にする必要があります。
- OpenType Stylistic Set 20 を使うと、必要な字体と文字間隔を利用できます (この Stylistic Set は特に数字に役立ちます)。この Stylistic Set を適用するメリットは、Segoe UI のすべての太さとスタイルで、一貫したテキスト レンダリングが保証されることです。

Segoe UI を使っていない場合、カーニング (kern) と標準合字 (liga) のみを適用することをお勧めします。推奨されるフォントを使わない場合は、随意の合字を適用しないでください。

```
<p style="font-family: 'Segoe UI'; -ms-font-feature-settings: 'kern' 1, 'dlig' 1, 'ss20' 1, 'lig' 1;">
    6/16/2012<br />
    To: Simon Daniels<br />
    Please find enclosed five flashing baffles.
</p>
```

```
.note {
    font-family: 'Segoe UI';
    -ms-font-feature-settings: 'kern' 1, 'dlig' 1, 'ss20' 1, 'lig' 1
}
```

次の図に、OpenType 機能を適用する前と適用した後のテキストを示します。

適用前	適用後
	

2. Unicode の機能を活用する

プレーン ASCII テキストを避け、適切な文字を使うことは、テキストの外観を洗練させるために最も効果的な方法です。

- 直線状の引用符とアポストロフィを避ける
- 小文字の "x" を使わず、乗算記号を使う
- 時間の区切り文字にはコロンではなく比例記号を使う
- 適切なダッシュを使い、ハイフンやマイナス記号を使わない

ハイフンの誤った使い方で最も多いのは、範囲の表現です。半角ダッシュを使ってください。タイポグラフィをさらに改善するには、半角ダッシュの前後に Hair Space を使います。数字またはすべて大文字のテキストを 1 行に揃えるときには、半角ダッシュをわずかに上へ移動して、視覚的な文字の中心に合わせます。

ニュース フィードなどの制御できないコンテンツがアプリに表示される場合、基本的な文字列の検索と置き換えの手法を使えば、アプリの実行中にこのようなコンテンツを自動的に修正することができます。

次に、Unicode を使ってテキストを改善する方法の例を示します。

Unicode 記号なし	Unicode 記号あり	使用する Unicode コード ポイント
"A quote."	“A quote.”	**U+201C、U+201D**
grocer's special	grocer’s special	**U+2019**
12:01	12∶01	**U+2236**

2x4	2 × 4	**U+00D7**
1/2 price	½ price	**U+00BD**
And the rest is history...	And the rest is history…	**U+2026**
A-C	A–C	**U+2013、U+200A（Hair Space）**

Unicode には記号も数千個含まれており、そのうち 722 個は標準に最近加えられたものです。このような記号の多くを、アプリのテキストやビットマップの代わりに使うことができます。使用可能なすべての記号の一覧については、「Segoe UI Symbol アイコンのガイドライン」をご覧ください。

3. 文字の先頭文字を大文字にする

テキストによるコミュニケーションでは、文の先頭文字を大文字にする従来からの形式が世界的に受け入れられています。すべて大文字または小文字のテキスト スタイルが、大文字または小文字のいずれかのみを使う国際的な書記体系になることはありません。標準的な文の先頭文字を大文字にする形式に従うことで、アプリの対象ユーザーが広がります。

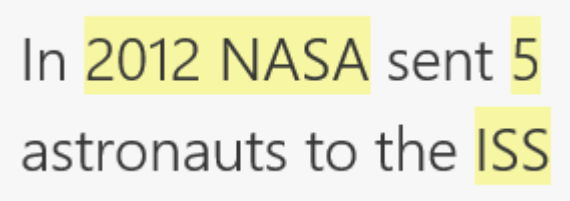
4. 特定の OpenType 機能を使う

最初の 3 つのルールに従えば、テキストの外観は非常に良くなるはずです。しかし、さらに上のレベルを目指すには、テキストの特定の部分に OpenType 機能を適用します。たとえば、頭字語には OpenType の "小型英大文字" 機能を適用します。この機能は本当の意味での小型英大文字を適用します (縮小した大文字ではありません)。コンテンツに数字が含まれる場合は、"旧スタイルの数字" 機能をテキストに適用します。この機能では、旧スタイルの数字 (小文字の数字と呼ばれる場合もあります) を使います。これがコンテンツに適している場合があります。

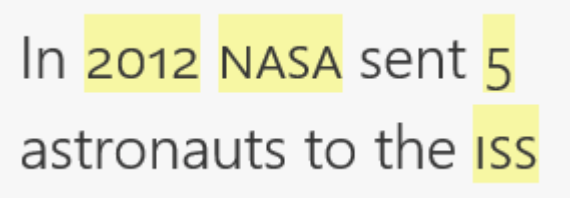
次に例を示します。

```
<p style="font-family: 'Segoe UI';" >
  In <span style="font-feature-settings: 'onum' 1;">2012</span> <span
style="font-feature-
  settings: 'c2sc' 1;">NASA</span> sent <span style="font-feature-settings:
'onum' 1;">5</span>
  astronauts to the <span style="font-feature-settings: 'c2sc' 1;">ISS</span>
</p>
```

旧スタイルの数字を適用する前:



後:



5. 編集する

編集は、よく見逃されるタイポグラフィの 1 面です。設計者の多くは、テキスト自体を変更する必要があるとは考えません。しかし、テキストが良い文章ではなく理解しづらければ、外観がどれほど良くても意味がありません。

6. カーニング、トラッキング、ハイフネーション、配置を効果的に使う

書体の設計者がよく言うことですが、文字の間隔は文字の形と同じくらい重要です。カーニング (最初のルールで説明しました) は、特定の 2 文字の組み合わせの間隔を調整することで、文字間隔を改善します。これはすべてのコンテンツに適用することをお勧めします。

一方で、トラッキングは文字列内のすべての文字の間に均等に余白を追加 (または削除) します。フォントの既定の文字間隔は特定のサイズでの読みやすさを考慮して最適化されているため、間隔をわずかに広げる方がよい場合もあります。トラッキングの値は .01 em または .02 em が便利です。

両端揃えのテキストを使うと、文字間隔を適切にすることが難しくなります。 両端揃えによってテキストを強制的にコンテナーに合わせると、常に文字間隔が犠牲になります。一般的に、テキストを両端揃えにすると、コンテンツ全体に空白の "川" ができます (単語間に大きなすきまができます)。両端揃えのテキストを避けて可能な限り左揃えを適用し、テキストの右端がでこぼこになりすぎないようにハイフネーションを適用してください。

7. フォント ベースのコントロールを使う

Windows 8 で最も知られていない秘密の 1 つは、UI で使われるボタン、アイコン、コントロールの多くは実際にはフォント ベースであることです。これはビットマップでもスケーラブル ベクター グラフィックス (SVG) でもなく、Segoe UI Symbol フォントの Unicode 私用領域に割り当てられたグリフです。さまざまなショーケース アプリでも同じ手法を使って、アイコンやコントロールを表示しています。

フォント ベースのアイコンとコントロールは通常のフォント文字と同様に縮小拡大できます。他のテキスト要素と並べて配置したり、レイヤーを使って色を付けることもできます。Windows 8.1 では、フォント ベースのコントロールの可能性をさらに高めるフルカラー フォントをサポートします。

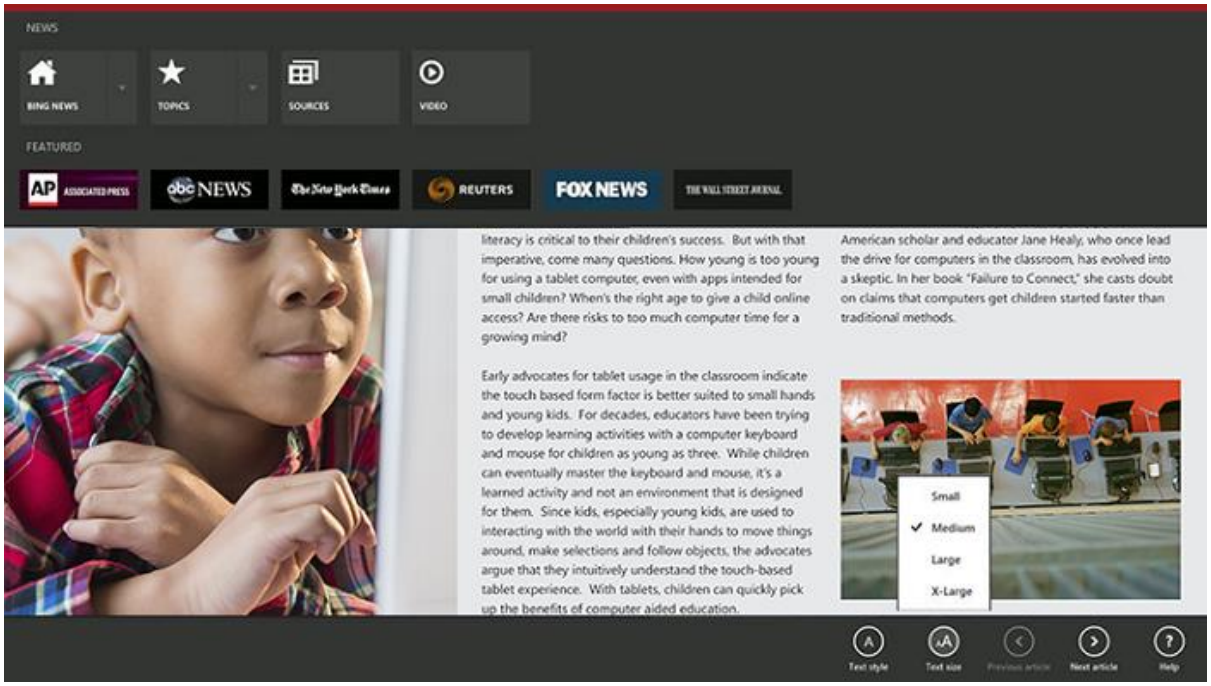
8. ユーザーが表示をカスタマイズできるようにする

ユーザーがシステム全体の設定を変更せずに本文のテキスト サイズを変更できる設定を用意します。テキスト サイズの値は、小 (11 ポイント)、中 (14 ポイント)、大 (18 ポイント) をお勧めします。さらに、明色の背景に暗色のテキスト (読みやすさに優れる) にするか、色付きの背景に暗色のテキストまたは暗色の背景に明色のテキスト (環境によっては眼精疲労の軽減に役立つ) にするかをユーザーがアプリで選べるようにすることもできます。ユーザーがさまざまな本文フォントを選べるようにすると、この機能を専用の電子書籍デバ

イスで使うユーザーに歓迎されるでしょう。種類を限定して、言語に適したフォントを用意してください。

リーダー アプリを作成する際には、下部のアプリ バーにユーザーがテキスト サイズを切り替えることができるフライアウトを追加することを検討してください。ユーザーに対して一貫性のある予測可能なエクスペリエンスを作るために、Segoe UI フォント サイズのアイコンを使えます。

以下の図は、アプリ バーにテキスト サイズの変更に使うフライアウトがあるニュース アプリです。



参考図書

ここで説明した基本的なアドバイスに従うことで、アプリのタイポグラフィは洗練されたものになるでしょう。しかし、ここでは最新のタイポグラフィと OpenType の機能を利用して実現できることのほんの一部を紹介しただけです。興味があれば、次の書籍を読むことをお勧めします。

- The Elements of Typographic Style (Robert Bringhurst 著)
- Inside Paragraphs: Typographic Fundamentals (Cyrus Highsmith 著)
- Detail in Typography (Jost Hochuli 著)

- Thinking with Type (Ellen Lupton 著)
- Stop Stealing Sheep (Erik Spiekermann および E.M. Ginger 著)

フォントの全要素の使用に関する IEBlog 記事は、カスケード スタイル シート (CSS) OpenType 機能を適用する方法を知るための入門としてお勧めです。ここには Monotype、FontShop、Font Bureau によるデモのリンクがあり、このテクノロジを使ってどのようなことができるかを知ることができます。

ユーザー操作

このセクションのガイドラインでは、ユーザーがアプリを操作できるさまざまな方法と、使いやすいアプリを設計するための推奨事項について説明します。

このセクションの内容

トピック	説明
アクセシビリティ	こうしたアクセシビリティ対応の設計の原則に従うことで、可能な限り広範な対象ユーザーにアプリを利用してもらい、Windows ストアでアプリに対してより多くのユーザーの興味を喚起するうえで役立ちます。
クロスライド	クロススライドは、スワイプ ジェスチャによる選択や、スライド ジェスチャによるドラッグ (移動) 操作をサポートするために使います。
光学式ズームとサイズ変更	このトピックでは、新しい Windows UI のズームと要素のサイズ変更について説明し、Windows ストア アプリでこのような新しい操作のメカニズムを使うときのユーザー エクスペリエンスのガイドラインを示します。
パン	パンとスクロールにより、ユーザーは単一ビュー内で移動し、ビューポートに収まらないビューのコンテンツを表示できます。ビューの例として、コンピューターのフォルダー構造、ドキュメントのライブラリ、フォト アルバムなどがあります。
回転	このトピックでは、新しい Windows UI の回転について説明し、Windows ストア アプリでこの新しい操作のメカニズムを使うときに考慮する必要があるユーザー エクスペリエンスのガイドラインを示します。
テキストと画像の選択	このトピックでは、テキスト、画像、コントロールを選んだり操作したりするための新しい Windows UI について説明します。また、Windows ストア アプリでこの新しい選択と操作のメカニズムを使うときに考慮する必要のあるユーザー エクスペリエンスに関するガイドラインを示します。

音声機能の設計	Windows Phone では、ユーザーは音声でアプリを操作できます。アプリに統合できる音声コンポーネントには、音声コマンド、音声認識、音声合成(TTS: text-to-speech) の 3 つがあります。
ターゲット設定	このトピックでは、タッチ補正のための接触形状の使用について説明し、Windows ランタイム アプリでのターゲット設定のベスト プラクティスを紹介します。
タッチ操作	タッチ用に最適化される一方で、さまざまな入力デバイスで一貫した機能を提供する、直観的で独特なユーザー操作エクスペリエンスを備えた Windows ストア アプリを作成します。
タッチ キーボード	タッチ キーボードを使うと、タッチをサポートしていて組み込みまたは外付けのキーボードを持たないデバイス上の Windows ストア アプリでもテキストを入力できます。
ビジュアルなフィードバック	ビジュアルなフィードバックは、Windows ストア アプリの対話操作が検出、解釈、処理されていることをユーザーに示すために使います。

アクセシビリティのガイドライン

アプリを設計する際は、ユーザーによってできる操作、できない操作、好ましい操作が大きく異なることを常に心に留めておいてください。こうしたアクセシビリティ対応の設計の原則に従うことで、可能な限り広範な対象ユーザーにアプリを利用してもらい、Windows ストアでアプリに対してより多くのユーザーの興味を喚起するうえで役立ちます。

アクセシビリティのための計画が必要な理由

アクセシビリティを考慮して設計したアプリは、次のシナリオに適しています。

- **画面の読み上げ:** 視覚に障碍があるユーザーは、アプリの UI の画像を頭の中に描いて保つために、スクリーン リーダーを使います。UI 要素の名前など、UI に関する情報を耳で聞くことで、UI コンテンツを把握し、使える機能を呼び出せるようになります。

  画面の読み上げをサポートするには、名前、役割、説明、状態、値など、UI 要素に関する正確な情報を十分にアプリで提供する必要があります。詳しくは、UI 要素に関する基本情報の公開についてのページ (HTML の場合または XAML の場合) をご覧ください。

  また、JavaScript と HTML を使った Windows ストア アプリにおけるライブ領域など、動的コンテンツを含む UI 要素に関する追加のアクセシビリティ情報を提供する必要もあります。スクリーン リーダーで追加のアクセシビリティ情報を使うことで、コンテンツに加えられた変更をユーザーに知らせることができます。HTML でライブ領域に関するアクセシビリティ情報を提供するには、動的コンテンツを含む要素の aria-live 属性を設定します。 詳しくは、「ライブ領域をアクセシビリティ対応にする」をご覧ください。XAML でライブ コンテンツの ARIA メタファーを使ってライブ領域のアクセシビリティ情報を提供するには、AutomationProperties.LiveSetting 添付プロパティを使います。
- **キーボードのアクセシビリティ:** キーボードはスクリーン リーダーを使ううえで不可欠であり、キーボードを使った方がアプリを効率よく操作できるユーザーにとっても重要です。キーボードでアクセスできるアプリでは、ユーザーはすべての対話型 UI 要素にキーボードだけでアクセスでき、次の操作を実行できます。
  - Tab キーと方向キーを使って、アプリ内を移動する。

- Space キーと Enter キーを使って、UI 要素をアクティブ化する。
- キーボード ショートカットを使って、コマンドとコントロールにアクセスする。

物理的なキーボードが存在しないシステムや、運動障碍により一般的な物理入力デバイスを使えないユーザーのために、スクリーン キーボードが用意されています。

詳しくは、キーボードのアクセシビリティの実装についてのページ (HTML の場合または XAML の場合) をご覧ください。

- **アクセシビリティに対応した視覚効果:** 視覚に障碍があるユーザーには、テキストを高いコントラスト比で表示する必要があります。また、UI がハイ コントラスト モードで適切に表示され、[コンピューターの簡単操作] コントロール パネルで **[画面上のすべてのものを大きくする]** を選んだときに適切に拡大されることも必要です。色を使って情報を伝える場合、色覚に障碍があるユーザーに対しては、色の代わりにテキスト、図形、アイコンなどを使う必要があります。詳しくは、ハイ コントラスト テーマのサポートについてのページ (HTML の場合または XAML の場合) をご覧ください。詳しくは、アクセシビリティに対応したテキストの要件の適合についてのページ (HTML の場合または XAML の場合) をご覧ください。

推奨と非推奨

- 名前、役割、説明、状態、値を含めて、アプリの UI 要素について情報を提供することにより、画面の読み上げをサポートします。
- ユーザーが Tab キーと方向キーを使ってアプリをナビゲートできるようにします。
- Space キーと Enter キーを使って、UI 要素をアクティブ化する。
- キーボード ショートカットを使って、コマンドとコントロールにアクセスする。
- ハイ コントラスト テーマをサポートするようにテキストと UI を設計します。
- **[簡単操作]** 設定が変更されたときに適切なスケーリングの調整を行うためのテキストおよび UI を準備します。
- 色を情報を伝える唯一の手段として使わないようにします。 色覚に障碍があるユーザーは、色によるステータス インジケーターのような色を通じてのみ伝えられ

る情報は受け取ることができません。他の視覚的な合図 (テキストが望ましい) を含めるようにして、情報にアクセスできるようにします。

- 1 秒間に 4 回以上光る UI 要素は使わないようにします。 明滅する要素は一部の人にとって発作を起こす原因になります。明滅する UI 要素の使用は避けた方が良いでしょう。
- ユーザー コンテキストを変えたり自動的に機能をアクティブ化しないようにします。 コンテキストやアクティブ化の変更は、フォーカスのある UI 要素上でユーザーが直接操作したときにだけ行うようにします。 ユーザー コンテキストの変更には、フォーカスの変更、新しいコンテンツの表示、別のページへの移動が含まれます。ユーザーと無関係に行われるコンテキストの変更は、障碍のあるユーザーを混乱させる可能性があります。 この要件の例外としては、サブメニューの表示、フォームの検証、別のコントロールでのヘルプ テキストの表示、非同期イベントへの応答によるコンテキストの変更などがあります。
- Windows ランタイムに含まれる既定のコントロール、または Microsoft UI オートメーション サポートを既に実装しているコントロールを使うことができる場合には、カスタム UI 要素を作成しないようにします。Windows ランタイムの標準コントロールは、既定でアクセシビリティに対応しており、追加する必要があるのは、通常、アプリ固有のわずかなアクセシビリティ属性のみです。それに対し、純粋なカスタム コントロールに **AutomationPeer** サポートを実装するのは、これより複雑です (「カスタム オートメーション ピア」をご覧ください)。
- 静的テキストなどの非対話型の要素をタブ オーダーに含めないようにします (たとえば、対話的に操作できない要素に **TabIndex** プロパティを設定することは避けます)。非対話型の要素をタブ オーダーに含めると、キーボード ナビゲーションの効率が下がるため、キーボードのアクセシビリティ ガイドラインで推奨されていません。多くの支援技術では、支援技術のユーザーにアプリのインターフェイスを表示する方法に関するロジックの一部として、要素をフォーカスする機能とタブ オーダーが使われています。タブ オーダーにテキスト専用要素が含まれると、タブ オーダーに対話型の要素 (ボタン、チェック ボックス、テキスト入力フィールド、コンボ ボックス、リストなど) しか含まれていないと想定しているユーザーを混乱させる可能性があります。
- UI 要素の絶対配置 (**Canvas** 要素内など) は、しばしば表示の順序が (事実上の論理的な順序である) 子要素の宣言の順序と異なるため、使用を避けます。 スクリーン

リーダーが UI 要素を正しい順序で読み取ることができるように、これらの要素をできるだけドキュメントの順序または論理的な順序に並べます。UI 要素の視覚的な順序がドキュメントの順序または論理的な順序とは異なる可能性がある場合は、正しい読み取り順序を定義するために明示的なタブ インデックス値 (**TabIndex** に設定) を使います。

- アプリの機能に本当に必要でない限り、アプリのキャンバス全体を自動的に更新しないようにします。 ページの内容を自動的に更新する必要がある場合には、ページの一部の領域だけを更新するようにします。支援技術では通常、更新されたアプリのキャンバスは、実質的な変更がわずかな場合でも、完全に新しい構造であると見なす必要があります。このため、更新されたアプリのドキュメント ビューや説明を再作成し、支援技術を使うユーザーに再提示する必要があります。

  注 領域内のコンテンツを更新する場合は、要素上の **AccessibilityProperties.LiveSetting** アクセシビリティ プロパティを既定以外の設定である **Polite** または **Assertive** に設定することをお勧めします。支援技術の中には、ライブ領域の Accessible Rich Internet Applications (ARIA) 概念にこの設定をマップして、コンテンツの領域が変更されたことをユーザーに通知できるものもあります。

  注 ユーザーが開始する意図的なページのナビゲーションによってアプリの構造が更新されることは、問題ありません。ただし、ナビゲーションを開始する UI 項目に適切な ID または名前を設定し、呼び出すとコンテキストが変更されてページが再読み込みされることがわかるようにしてください。

その他の使い方のガイダンス

HTML カスタム コントロールのアクセシビリティ対応

HTML カスタム コントロールを使う場合、アクセシビリティに対応する名前、役割、状態、値など、コントロールに関する基本的なアクセシビリティ情報をすべて用意しておく必要があります。また、コントロールがキーボードで完全にアクセスでき、UI が視覚に関するアクセシビリティの要件を満たしていることを確認する必要もあります。

たとえば、カスタムの対話要素を表す (つまり、**onclick** イベントを処理する) **div** 要素を含めるとします。 次の操作を行う必要があります。

- **div** 要素のアクセシビリティ対応の名前を設定します。
- role 属性を、Accessible Rich Internet Applications (ARIA) の対応する対話型の役割 ("button" など) に設定します。
- tabindex 属性を、タブの順序で要素を含むように設定します。
- キーボードによるアクティブ化をサポートする keyboard イベント ハンドラーを追加します。これは **onclick** イベント ハンドラーのキーボード版です。

アクセシビリティのためのカスタム HTML UI 要素の公開について詳しくは、「Accessible Rich Internet Applications (ARIA)」をご覧ください。

注 HTML5 の **canvas** 要素では、アクセシビリティがサポートされません。**canvas** にはコンテンツのアクセシビリティ情報を公開する方法がないため、どうしても必要な場合以外は、使わないでください。**canvas** を使う場合は、カスタム UI 要素として扱います。

XAML カスタム コントロールのアクセシビリティ対応

XAML カスタム コントロールを使う場合、アクセシビリティに対応する名前、役割、状態、値など、コントロールに関する基本的なアクセシビリティ情報の一部を調整することが必要になる可能性があります。また、コントロールがキーボードで完全にアクセスでき、UI が視覚に関するアクセシビリティの要件を満たしていることを確認する必要もあります。XAML 用のカスタム コントロールを作成する場合、カスタム コントロールの基底クラスとして使用する、あらゆるコントロールで使用できる UI オートメーション サポートを継承します。これで十分な場合もあります。コントロールをカスタマイズする度合いに応じて、既定の UI オートメーション サポートを変更または強化する、カスタム UI オートメーション ピア クラスを作成することもできます。これは、Windows.UI.Xaml.Automation 名前空間と Windows.UI.Xaml.Automation.Peers 名前空間の API によって実現されます。詳しくは、「カスタム オートメーション ピア」をご覧ください。

開発プラットフォームでのアクセシビリティのサポート

Windows Runtime の開発プラットフォームでは、開発サイクルのすべての段階でアクセシビリティをサポートしています。

- **作成:** Microsoft Visual Studio のアプリ テンプレートから生成されたコードには、アクセシビリティ情報が含まれます。
- **コード作成:** 開発プラットフォームでは、コード作成段階で次のアクセシビリティをサポートしています。◦Visual Studio の Microsoft IntelliSense には、アクセシビリティ情報が含まれています。
  - ◦ Windows 8 プラットフォームに含まれているコントロールには、アクセシビリティのサポートが組み込まれています。標準の HTML とプラットフォームのコントロールを使うと、ほとんどのアクセシビリティを既定のプラットフォームの動作としてサポートできます。たとえば、評価コントロールは完全にアクセシビリティに対応しており、追加の作業は必要ありません。また、**ListView** コントロールは、メインのリスト要素のアクセス可能な名前だけが必要です。それ以外のアクセシビリティのサポートはすべて組み込まれています。プラットフォーム コントロールの一覧については、「コントロールの一覧 (HTML)」または「コントロールの一覧 (XAML)」をご覧ください。
  - ◦ Windows デベロッパー センターのドキュメントには、アクセシビリティのガイドラインとサンプル アプリが含まれています。
- **テスト:** Windows ソフトウェア開発キット (Windows SDK) には、アクセシビリティのテスト ツールが用意されています。詳しくは、アプリのアクセシビリティのテストについてのページ (HTML の場合または XAML の場合) をご覧ください。
- **販売:** Windows ストアではアプリを公開する際にアクセシビリティ対応として登録できます。そうすると、ユーザーが Windows ストアでアクセシビリティ フィルターを使ってアプリを見つけられるようになります。 詳しくは、Windows ストアでアプリをアクセシビリティ対応として宣言する方法についてのページ (HTML の場合または XAML の場合) をご覧ください。

XAML カスタム コントロールのアクセシビリティ対応

XAML カスタム コントロールを使う場合、アクセシビリティに対応する名前、役割、状態、値など、コントロールに関する基本的なアクセシビリティ情報の一部を調整することが必要になる可能性があります。また、コントロールがキーボードで完全にアクセスでき、UI が視覚に関するアクセシビリティの要件を満たしていることを確認する必要もあります。XAML 用のカスタム コントロールを作成する場合、カスタム コントロールの基底クラスとして使用する、あらゆるコントロールで使用できる Microsoft UI オートメーション サポートを継承します。これで十分な場合もあります。コントロールをカスタマイズする度合いに応じて、既定の UI オートメーション サポートを変更または強化する、カスタム UI オートメーション ピア クラスを作成することもできます。これは、Windows.UI.Xaml.Automation 名前空間と Windows.UI.Xaml.Automation.Peers 名前空間の API によって実現されます。詳しくは、「カスタム オートメーション ピア」をご覧ください。

クロススライドのガイドライン

クロススライドは、スワイプ ジェスチャーによる選択や、スライド ジェスチャーによるドラッグ (移動) 操作をサポートするために使います。

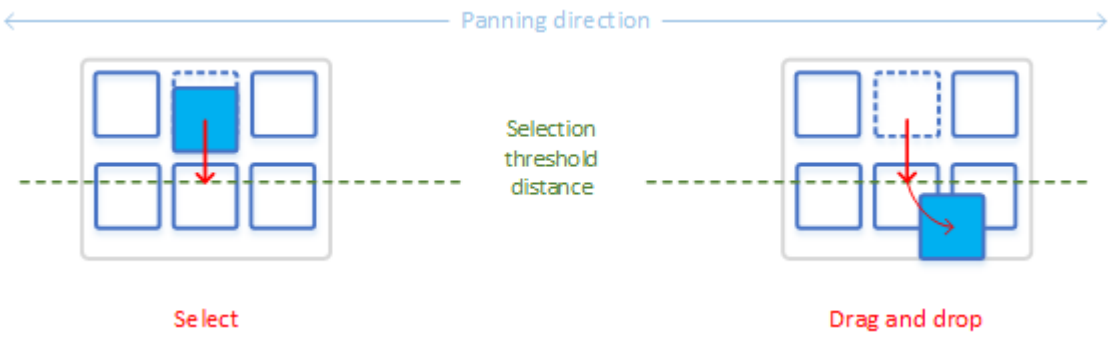
推奨と非推奨

- クロススライドは、単一の方向にスクロールするリストやコレクションだけに使います。
- クロススライドは、タップ操作が別の目的で使われる場合に、項目を選ぶために使います。
- キューに項目を追加するためにクロススライドを使わないでください。

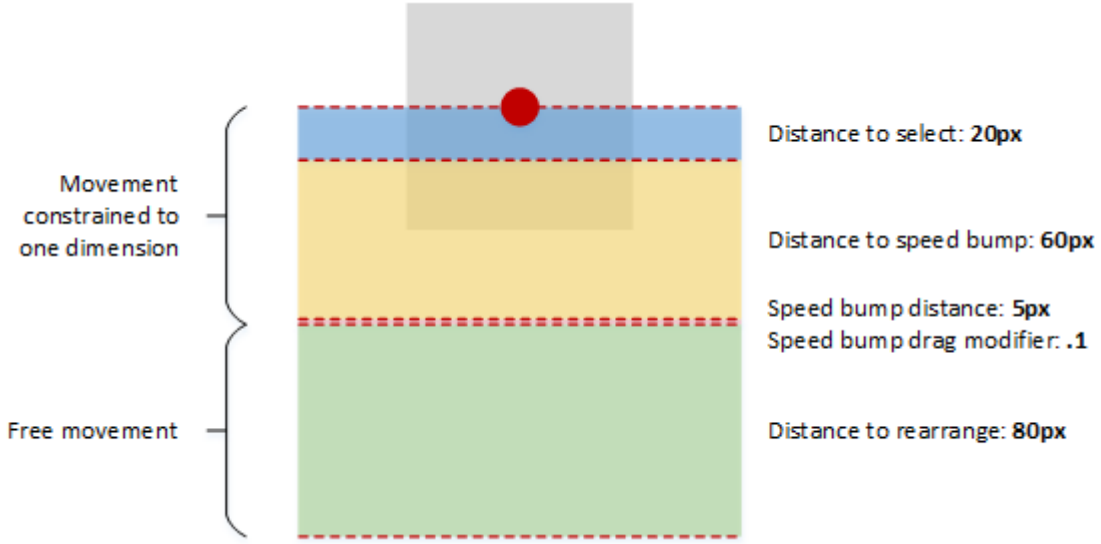
その他の使い方のガイダンス

選択とドラッグは、1 方向 (垂直または水平) にパンできるコンテンツ領域内でだけ行うことができます。これらの操作が機能するには、1 つのパン方向がロックされていて、ジェスチャーがパン方向に対して垂直な方向に行われる必要があります。

ここでは、クロススライドを使ってオブジェクトを選び、ドラッグする方法を示します。左の図は、スワイプ ジェスチャーで距離のしきい値を超える前に指を離してオブジェクトを解放することで、項目が選択された状態を示しています。右の図は、距離のしきい値を超えてスライド ジェスチャーを行うことで、オブジェクトがドラッグされた状態を示しています。

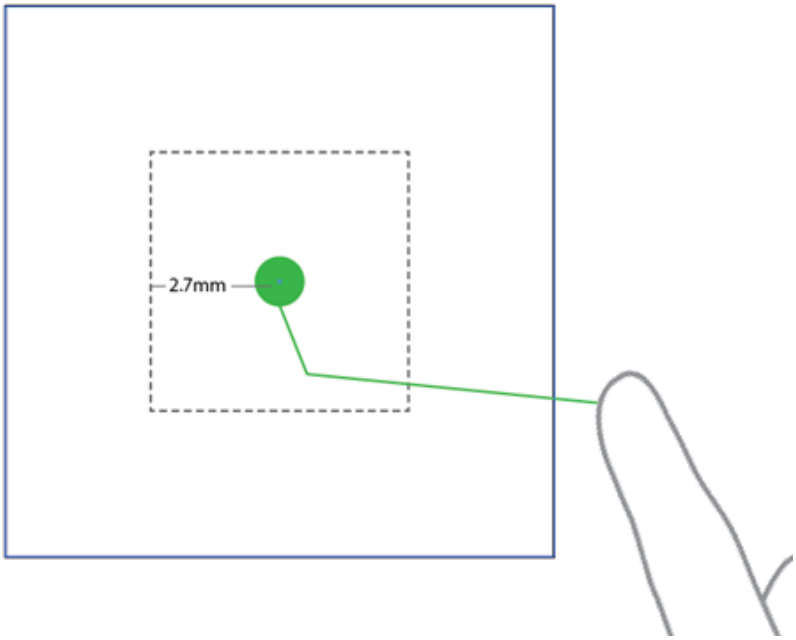


クロススライド操作で使われるしきい値の距離を次の図に示します。



パンの機能を維持するために、選択操作とドラッグ操作は、2.7 mm (ターゲット解像度で約 10 ピクセル) という小さいしきい値を超えないと有効にならないしくみになっています。この小さいしきい値は、クロススライドとパンの区別に使われるほか、タップ ジェスチャーをクロススライドやパンと区別する目的でも使われます。

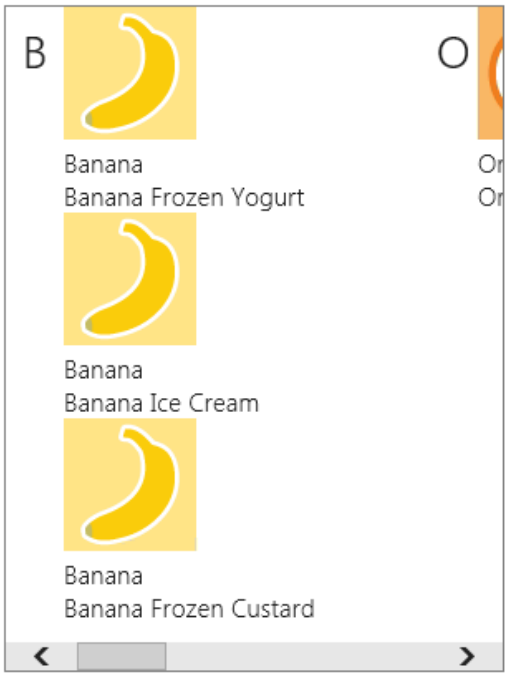
この図は、ユーザーが UI の要素にタッチしたときに、指の位置がわずかに下に動いて しまった状態を示しています。 しきい値がなければ、最初に垂直方向に 移動しているため、この操作はクロススライドと解釈されてしまいます。このしきい値があるおかげで、水平方向のパンと正しく解釈されます。

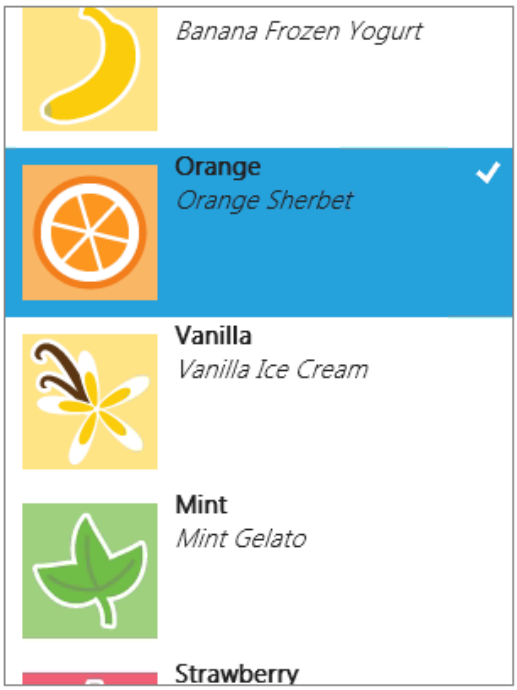


次に、クロススライド機能を Windows ストア アプリに含める際に考慮する必要がある、いくつかのガイドラインを示します。

クロススライドは、単一の方向にスクロールするリストやコレクションだけに使います。

注 Web ブラウザーや電子ブック リーダーのように、コンテンツ領域を 2 方向にパンできる場合は、時間制限のある長押し操作を使って、画像やハイパーリンクなどのオブジェクトのコンテキスト メニューを呼び出すようにしてください。





水平方向にパンする 2 次元のリスト。 項目を選択または移動するには垂直方向にドラッグします。	垂直方向にパンする 1 次元のリスト。項目を選択または移動するには水平方向にドラッグします。

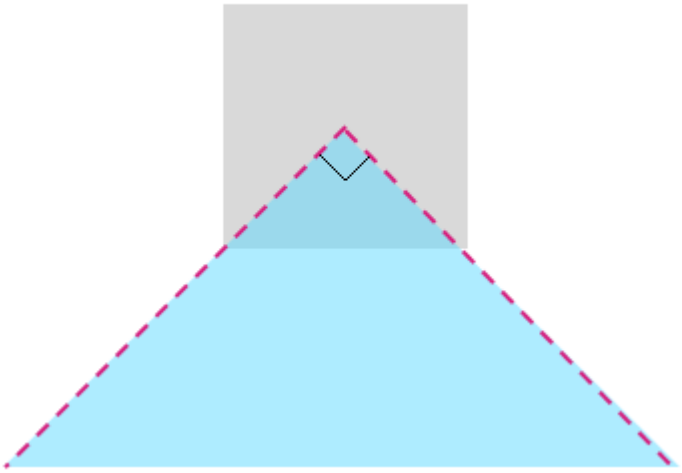
選択

選択は、1 つ以上のオブジェクトを起動またはアクティブ化せずにマークする操作です。これは、マウスを 1 回クリックする操作、または Shift キーを押しながらクリックする操作 (オブジェクトが複数の場合) に相当します。

クロススライド選択を行うには、要素をタッチし、少しドラッグして放します。この選択方法を使えば、他のタッチ インターフェイスで必要になるような、専用の選択モードや時間

制限のある長押し操作は必要ありません。また、アクティブ化のためのタップ操作と競合することもありません。

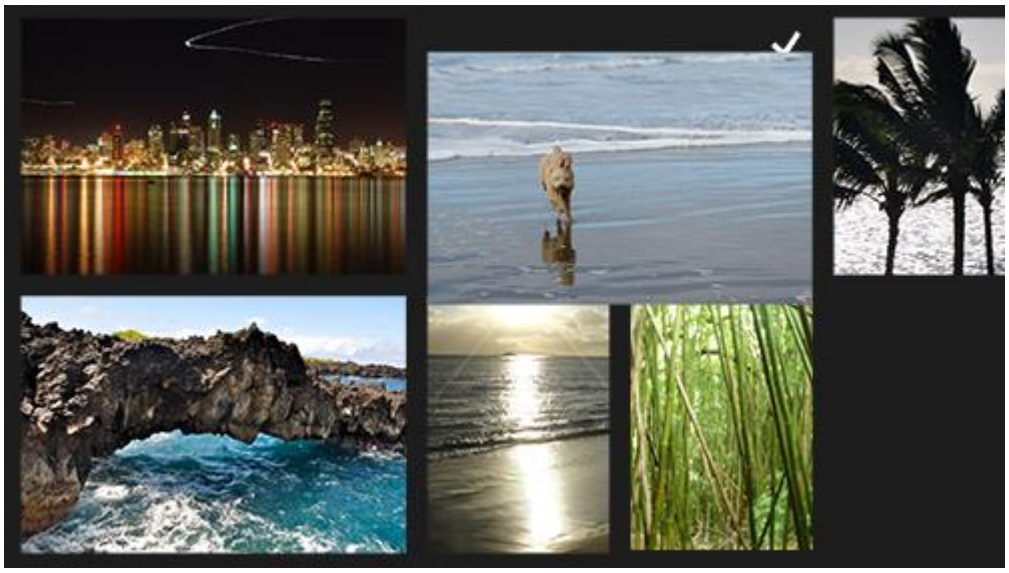
クロススライド選択には、距離のしきい値のほかにも領域のしきい値があり、次の図に示すように範囲が 90° に制限されます。この領域の外にドラッグすると、オブジェクトは選択されません。



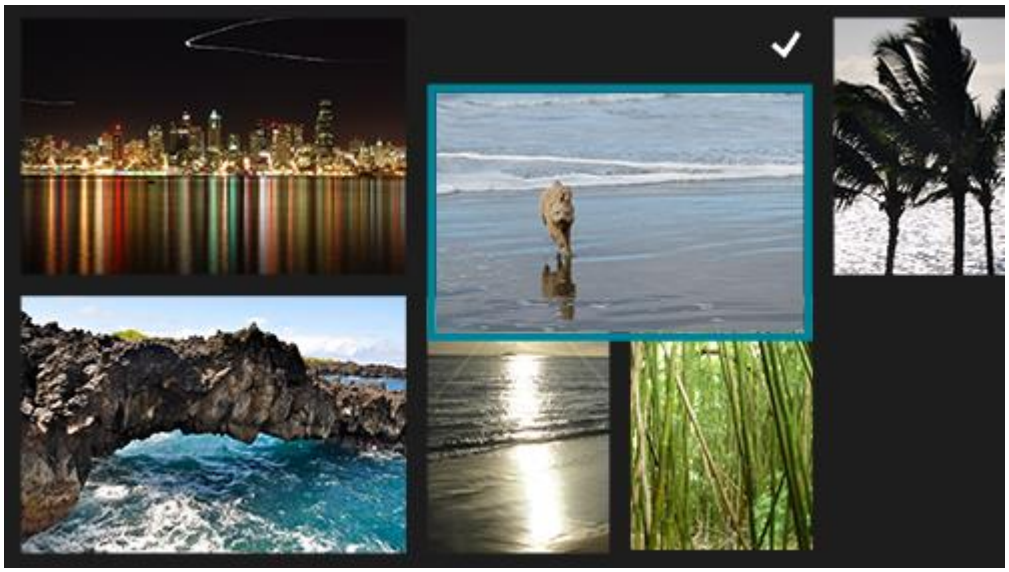
クロススライド操作を補完する操作に、"自己表明" 操作とも呼ばれる時間制限のある長押し操作があります。この補助操作でアクティブ化されるアニメーションによって、オブジェクトに対して実行できる操作が示されます。

次のスクリーン ショットは、自己表明操作のアニメーションの動作を示しています。

1. 長押しして、自己表明操作のアニメーションを開始します。項目が選ばれているかどうかによって、アニメーションで説明される内容が変わります。選ばれていない場合はチェック マークが付き、選ばれている場合はチェック マークが付きません。



2. スワイプ ジェスチャ —(上または下) を使って項目を選びます。



3. この時点で、項目が選ばれています。スライド ジェスチャーを使って選択動作を上書きし、項目を移動します。

主な操作が選択だけであるアプリケーションでは、選択にシングル タップを使います。この場合、アクティブ化やナビゲーションのための標準のタップ操作と区別するために、クロススライドの自己表明のアニメーションが表示されます。

選択バスケット

選択バスケットは、アプリの主要なリストやコレクションから選択された項目を視覚的に区別して動的に表す機能です。これは選択された項目の追跡に役立つ機能で、次のようなアプリで使うと便利です。

- 項目を複数の場所から選択できる。
- 複数の項目を選択できる。
- 選択リストによって操作やコマンドが異なる。

選択バスケットの内容は、操作やコマンドの実行後も保持されます。たとえば、ギャラリーから一連の写真を選択して各写真に色補正を適用し、それらの写真を何らかの方法で共有した場合、それらの項目は選択されたままになります。

アプリで選択バスケットを使わない場合は、操作やコマンドの実行後に現在の選択がクリアされます。たとえば、再生リストから曲を選択して評価した後、その選択はクリアされるなどです。

また、選択バスケットを使わない場合は、リストやコレクションで別の項目がアクティブ化されたときにも現在の選択がクリアされます。たとえば、受信トレイのメッセージを選択すると、プレビュー ウィンドウが更新されます。その後、受信トレイで別のメッセージを選択すると、前のメッセージの選択が取り消され、プレビュー ウィンドウが更新されることなどです。

キュー

キューと選択バスケットのリストは異なるものであるため、混同しないように注意してください。主な違いは次のとおりです。

- 選択バスケットの項目のリストは、視覚的に表すことだけを目的としたものです。キューの項目は、特定の操作を想定してまとめられたものです。
- 選択バスケットでは同じ項目は 1 回しか表示できませんが、キューでは複数回表示できます。
- 選択バスケットの項目の順序は、選択の順序を表します。 キューの項目の順序は、機能に直接関連します。

これらの理由から、キューに項目を追加する目的でクロススライド選択操作を使わないでください。 キューに項目を追加するときは、代わりにドラッグ操作を使います。

ドラッグ

1 つまたは複数のオブジェクトを別の場所に移動するには、ドラッグを使います。

複数のオブジェクトを移動する必要がある場合は、ユーザーが複数の項目を選択してから、すべてを同時にドラッグできるようにします。

光学式ズームとサイズ変更のガイドライン

このトピックでは、新しい Windows UI のズームと要素のサイズ変更について説明し、Windows ストア アプリでこのような新しい操作のメカニズムを使うときのユーザー エクスペリエンスのガイドラインを示します。

推奨と非推奨

サイズ変更または光学式ズームをサポートするアプリでは、次のガイドラインに従ってください。

- 最大サイズと最小サイズの制限または範囲が定義されている場合には、ビジュアルなフィードバックを使って、ユーザーがこの制限に達したことや超過したことを示します。
- スナップ位置を使うと、論理的な操作停止位置を指定してズームとサイズ変更の動作を変更し、コンテンツの特定の部分がビューポートに表示されるようにできます。一般的なズーム レベルまたは論理ビューに対してスナップ位置を設定して、ユーザーがこれらのレベルを簡単に選べるようにします。たとえば、写真のアプリでは 100% の位置にサイズ変更用のスナップ位置を設定します。また、地図のアプリでスナップ位置を設定すると、市、県、国を表示する場合に便利です。
  スナップ位置があると、ユーザーの操作が正確でなくても意図された操作を実行できます。
  スナップ位置には次の 2 種類があります。
  - 近接: 指を離した後、スナップ位置の距離のしきい値の範囲内で慣性に従った動きが止まると、スナップ位置が選ばれます。近接スナップ位置の場合は、ズームとサイズ変更をスナップ位置とスナップ位置の間で止めることができます。
  - 強制: 指を離す前に通過した最後のスナップ位置の直前または直後のスナップ位置が選ばれます (ジェスチャの方向と速度によって異なります)。操作が必ず強制スナップ位置で止まるようにする必要があります。
- 慣性の法則を使う必要があります。これには次のものがあります。
  - 減速: ユーザーが 2 本の指を互いに近づけたり、遠ざけたりしたときに発生します。これは滑りやすい表面で滑っている状態から止まるまでの動きに似ています。

- バウンド: サイズの制限または範囲を超えると、わずかな跳ね返りの効果が発生します。
- 「ターゲットの設定のガイドライン」に従った領域制御。
- UI の操作またはアプリ内の追加コントロールの公開用にはズームを使わず、パン領域を使います。パンについて詳しくは、「パンのガイドライン」をご覧ください。

サイズ変更をサポートするアプリでは、次のガイドラインに従ってください。

- 制限付きのサイズ変更のためにスケーリング ハンドルを提供します。ハンドルが指定されない場合は、等角投影、つまり比が一定のサイズ変更が既定値です。
- サイズ変更できるコンテンツ領域内にサイズ変更できるオブジェクトを置かないようにします。ただし、次のような例外があります。
  - サイズ変更できるアイテムがサイズ変更できるキャンバスまたはアート ボードに表示される描画アプリケーション。
  - 地図などの埋め込みオブジェクトがある Web ページ。

注　どのような場合でも、すべてのタッチ ポイントがサイズ変更できるオブジェクト内にある場合以外は、コンテンツ領域のサイズが変更されます。

その他の使い方のガイダンス

光学式ズームを使うと、ユーザーはコンテンツの表示を拡大できます (コンテンツ領域自体に対して実行されます)。一方、サイズ変更を使うと、コンテンツ領域に対する表示は変更せずに、1 つまたは複数のオブジェクトの相対的なサイズをユーザーが変更できます (コンテンツ領域内のオブジェクトに対して実行されます)。

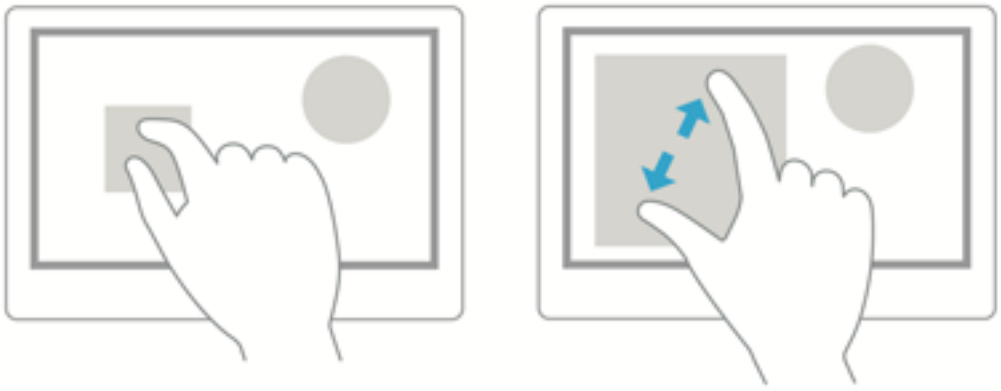
注　光学式ズームとセマンティック ズーム操作を混同しないように気を付けてください。これらは同じジェスチャーを共有しますが、セマンティック ズームは、構造化されたデータまたはコンテンツを単一のビュー内で表示したりナビゲーションしたりする場合に使われます (コンピューターのフォルダー構造、ドキュメント ライブラリ、フォト アルバムなど)。

光学式ズーム操作とセマンティック ズーム操作は両方とも、ピンチ ジェスチャーとストレッチ ジェスチャー (指を広げて拡大、互いに近づけて縮小)、Ctrl キーを押しながらマウスのスクロール ホイールをスクロール、または Ctrl キーを (テンキーがない場合は Shift キーも同時に) 押しながらプラス (+) キーまたはマイナス (-) キーを押して実行します。

次の図にサイズ変更と光学式ズームの違いを示します。

光学式ズーム: ユーザーは領域を選び、領域全体を拡大します。

サイズ変更: ユーザーは領域内のオブジェクトを選び、そのオブジェクトのサイズを変更します。



パンのガイドライン

パンとスクロールにより、ユーザーは単一ビュー内で移動し、ビューポートに収まらないビューのコンテンツを表示できます。ビューの例として、コンピューターのフォルダー構造、ドキュメントのライブラリ、フォト アルバムなどがあります。

推奨と非推奨

パン インジケーターとスクロール バー

- アプリにコンテンツを読み込む前に、パン/スクロールが可能であることを確認します。
- パン インジケーターとスクロール バーを表示して、位置とサイズがわかるようにします。 これらのコントロールは、カスタム ナビゲーション機能がある場合には非表示にします。

  注 標準のスクロール バーとは異なり、パン インジケーターは情報提供のみを目的としています。 入力デバイスには公開されず、一切操作できません。

単一軸パン (1 次元のオーバーフロー)

- コンテンツ領域が 1 つのビューポート境界 (垂直方向または水平方向) を超えている場合は、単一軸のパンを使います。
  - 1 次元の項目の一覧の場合は、垂直方向のパンを使います。
  - 項目のグリッドの場合は、水平方向のパンを使います。
- ユーザーのパン操作をスナップ位置以外の位置で停止できるようにする必要がある場合は、単一軸パンで強制スナップ位置を使わないでください。 強制スナップ位置を使うと、スナップ位置で必ず停止します。代わりに、近接スナップ位置を使ってください。

フリーフォーム パン (2 次元のオーバーフロー)

- コンテンツ領域が両方のビューポート境界 (垂直方向と水平方向) を超えている場合は、2 軸のパンを使います。
  - 複数の方向へ動かされる可能性がある、構造化されていないコンテンツの場合は、既定のレール動作を上書きしてフリーフォーム パンを使います。
- フリーフォーム パンは通常、画像や地図内の移動に適しています。

ページ ビュー

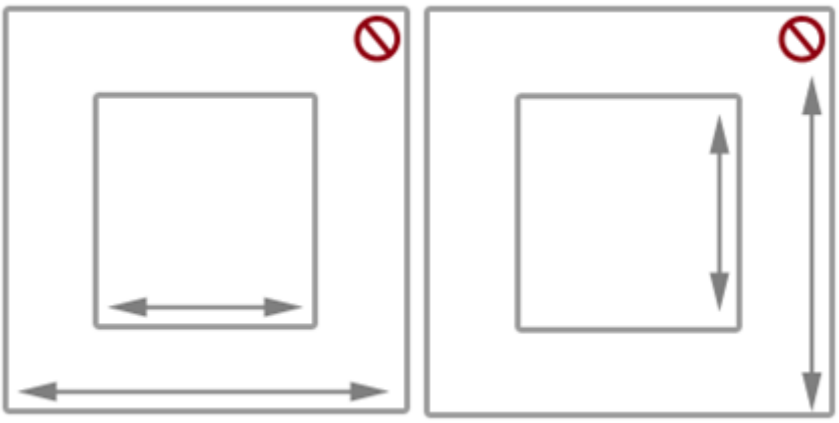
- コンテンツが個別の要素で構成されている場合、または 1 つの要素全体を表示する必要がある場合は、強制スナップ位置を使います。書籍や雑誌のページ、項目の列、個々の画像がその例です。
  - スナップ位置はそれぞれの論理的な境界に置く必要があります。
  - 各要素のサイズや倍率を、ビューに収まるように調整する必要があります。

論理的な位置と主要位置

- コンテンツ内にユーザーが停止する可能性が高い主要位置または論理的な位置がある場合は、近接スナップ位置を使います。たとえば、セクション ヘッダーなどです。
- 最大サイズと最小サイズの制限または範囲が定義されている場合には、視覚的なフィードバックを使って、ユーザーがこの制限に達したことや超過したことを示します。

埋め込まれたコンテンツまたは入れ子になったコンテンツの連結

- テキストとグリッド ベースのコンテンツに対して単一軸パン (通常は水平方向) と列レイアウトを使います。このような場合は、コンテンツは通常列から列へと自然に折り返し、遷移するので、Windows ストア アプリ全体で一貫性があり見つけやすいユーザー エクスペリエンスを維持できます。
- テキストまたは項目の一覧を表示する目的で、埋め込まれたパン対応領域を使わないでください。領域内で入力の接触が検出されたときしかパン インジケーターとスクロール バーが表示されず、直感的で見つけやすいユーザー エクスペリエンスが得られません。
- 下の図に示すように、2 つのパン対応領域がどちらも同じ方向にパンする場合は、パン対応領域を別のパン対応領域内に連結 (配置) しないでください。この場合に連結すると、子領域の境界に到達したときに親領域が意図せずパンされる可能性があります。パンの軸は相互に対して垂直になるようにしてください。



その他の使い方のガイダンス

タッチでのパン (1 本または複数の指でのスワイプまたはスライド ジェスチャー) は、マウスでのスクロールと似ています。パンはスクロール バーのクリックよりも、マウス ホイールの回転やスクロール ボックスのスライドに最も近い操作です。API で区別されているか、一部のデバイス固有の Windows UI によって区別が必要とされていない限り、両方の操作を単にパンと呼びます。

入力デバイスに応じて、ユーザーは次のいずれかを使って、パン対応領域内でパンを実行します。

- マウス、タッチパッド、またはアクティブなペン/スタイラスを使って、スクロール矢印をクリックするか、スクロール ボックスをドラッグするか、スクロール バー内をクリックする。
- マウスのホイール ボタンを使って、スクロール ボックスのドラッグと同じ動作を実現する。
- マウスでサポートされている場合は、拡張ボタン (XBUTTON1 と XBUTTON2)。
- キーボードの方向キーを使ってスクロール ボックスのドラッグと同じ動作を実現するか、ページ キーを使ってスクロール バー内のクリックと同じ動作を実現する。
- タッチ、タッチパッド、またはパッシブなペン/スタイラスを使って、任意の方向に指をスライドまたはスワイプする。

スライドでは、指をパン方向にゆっくり移動します。これにより、コンテンツが指と同じ速度で同じ距離だけパンする 1 対 1 の関係ができます。スワイプ (指をすばやくスライドして離す) では、パンのアニメーションに次の物理的効果が適用されます。

- 減速 (慣性): 指を離すとパンが減速し始めます。これは滑りやすい表面で滑っている状態から止まるまでの動きに似ています。
- 吸収: 減速時に、パン操作の勢いがスナップ位置またはコンテンツ領域の境界まで保たれた場合、反対方向に少し押し戻される効果があります。

パンの種類

Windows 8 では 3 種類のパンがサポートされます。

- 単一軸: 一方向 (水平または垂直) へのパンのみがサポートされます。
- レール: 全方向へのパンがサポートされます。ただし、特定の方向への距離のしきい値を超えると、パンはその軸に制限されます。
- フリーフォーム: 全方向へのパンがサポートされます。

パンの UI

パンの操作エクスペリエンスは、機能的には類似していても、入力デバイスごとに異なります。

パン対応領域

パン対応領域の動作は、JavaScript を使った Windows ストア アプリの開発者に対して、設計時にカスケード スタイル シート (CSS) を通じて公開されます。

検出された入力デバイスに基づいて、次の 2 種類のパン表示モードが使われます。

- パン インジケーター (タッチを使う場合)。
- スクロール バー (マウス、タッチパッド、キーボード、スタイラスなど、その他の入力デバイスを使う場合)。

注 パン インジケーターは、タッチによる接触がパン対応領域内であるときにだけ表示されます。同様に、スクロール バーは、スクロール対応領域内にマウス カーソル、ペン/スタイラス カーソル、またはキーボード フォーカスがあるときにのみ表示されます。

パン インジケーター

パン インジケーターは、スクロール バーのスクロール ボックスに似ています。パン対応領域全体に対する表示されているコンテンツの比率と、パン対応領域内の表示されているコンテンツの相対的な位置を示します。

次の図は、長さが異なる 2 つのパン対応領域とそれらのパン インジケーターを示しています。

パン の振る舞い

スナップ位置

パンとスワイプ ジェスチャーを使うと、タッチによる接触が離れたときの操作に慣性の動作が生じます。慣性によって、コンテンツのパンは、ユーザーによる直接入力がなければ距離のしきい値に到達するまで継続されます。この慣性の動作を変更するには、スナップ位置を使います。

スナップ位置は、アプリのコンテンツの論理的な停止を指定します。スナップ位置は、認識に基づくユーザー用のページング メカニズムとして機能し、ユーザーが大きなパン対応領域でスライドまたはスワイプしすぎて疲れるのを防ぎます。これらを使用すると、不正確なユーザー入力を処理し、コンテンツの特定の部分や主要な情報がビューポートに確実に表示されるようにすることができます。

スナップ位置には次の 2 種類があります。

- 近接: 指を離した後、スナップ位置の距離のしきい値の範囲内で慣性に従った動きが止まると、スナップ位置が選ばれます。パンは近接スナップ位置の中間で停止することもできます。
- 強制: 指を離す前に通過した最後のスナップ位置の直前または直後のスナップ位置が選ばれます (ジェスチャーの方向と速度によって異なります)。パンは強制スナップ位置で停止する必要があります。

  パンのスナップ位置は、ページ付けされたコンテンツと同じ動作を実現したり、項目を論理的にグループ化して、ビューポートまたはディスプレイに収まるように動的に再グループ化できるようにしたりする、Web ブラウザーやフォト アルバムのようなアプリで便利です。

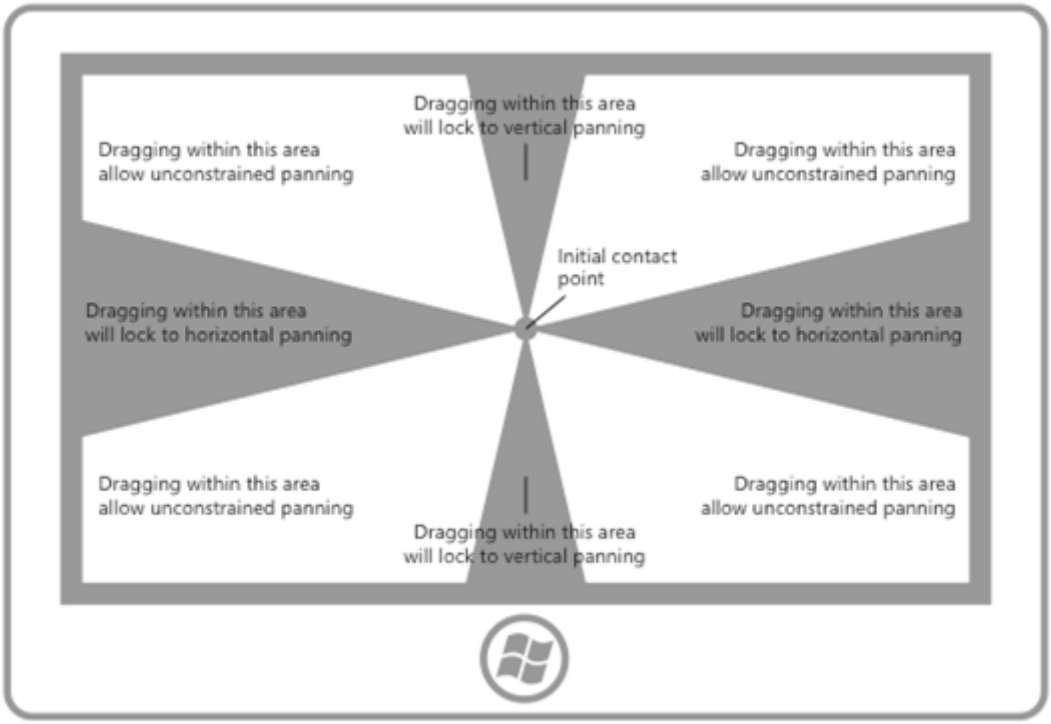
  次の図は、特定の位置にパンして離すことでコンテンツを論理的な位置に自動的にパンする方法を示しています。

スワイプしてパンします。	タッチによる接触を離します。	パン対応領域は、タッチによる接触が離れた場所ではなく、スナップ位置で停止します。

レール

コンテンツは、ディスプレイ デバイスのサイズと解像度より広かったり高かったりする場合があります。このため、2 次元のパン (水平方向と垂直方向) が必要になることがよくあります。レールは、このような場合に動作の主軸 (垂直方向または水平方向) に沿ってパンを強調表示することで、ユーザー エクスペリエンスを向上させます。

次の図は、レールの概念を示しています。

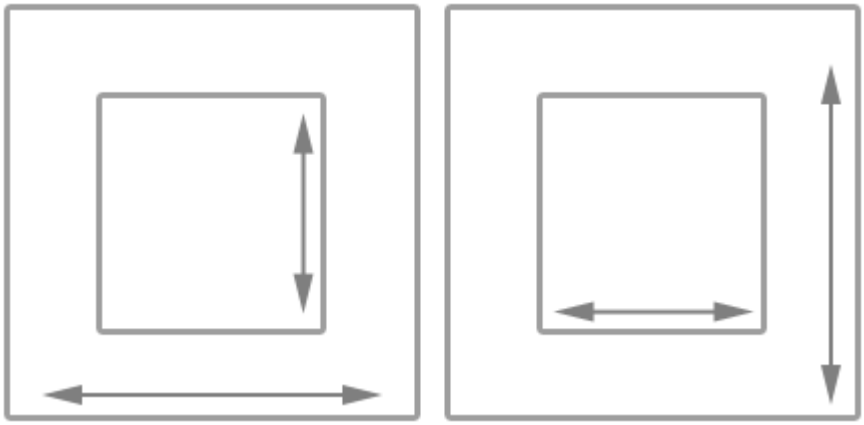


埋め込まれたコンテンツまたは入れ子になったコンテンツの連結

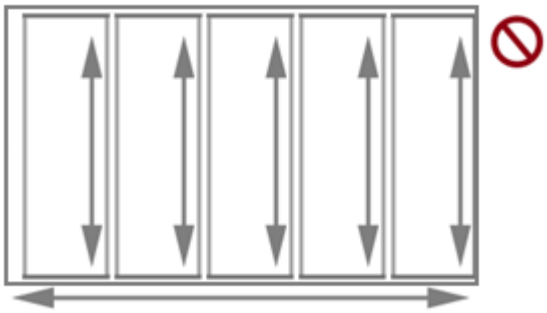
他のズーム可能またはスクロール可能な要素の入れ子になっている要素のズームまたはスクロールが限界に達した後で、親要素が子要素のズーム操作またはスクロール操作を継続して開始するかどうかを指定します。これはズームまたはスクロールのチェーンと呼ばれます。

1 つ以上の単一軸パン領域またはフリーフォーム パン領域が含まれる単一軸のコンテンツ領域内で (これらの子領域のいずれかでタッチによる接触があったときに) パンを行う場合は、連結を使います。子領域の特定の方向のパン境界に到達すると、親領域の同じ方向にパンがアクティブ化されます。

パン対応領域を別のパン対応領域内に入れ子にするときは、コンテナーと埋め込まれたコンテンツ間に十分な領域を指定することが重要です。次の図では、パン対応領域が別のパン対応領域内に置かれており、それぞれが相互に対して垂直方向に移動します。各領域にユーザーがパンできる十分な領域があります。



十分な領域がないと、次の図に示すように、埋め込まれたパン対応領域によってコンテナーでのパンが妨げられ、1 つ以上のパン対応領域で意図しないパンが発生する可能性があります。



このガイダンスは、たとえば、フォト アルバムや地図のようなアプリでも役に立ちます。各画像または地図内の制約のないパンをサポートしながら、アルバム内の前の画像または次の画像や詳細な領域への単一軸パンもサポートできます。フリーフォーム パンの画像や地図に対応する詳細領域またはオプション領域を提供するアプリでは、ページ レイアウトを詳細領域やオプション領域で始めることをお勧めします。画像や地図の制約のないパン領域が、詳細領域へのパンを妨げる可能性があるためです。

回転のガイドライン

このトピックでは、新しい Windows UI の回転について説明し、Windows ストア アプリでこの新しい操作のメカニズムを使うときに考慮する必要があるユーザー エクスペリエンスのガイドラインを示します。

推奨と非推奨

- ユーザーが直接 UI 要素を回転できるように回転を使います。

その他の使い方のガイダンス

回転の概要

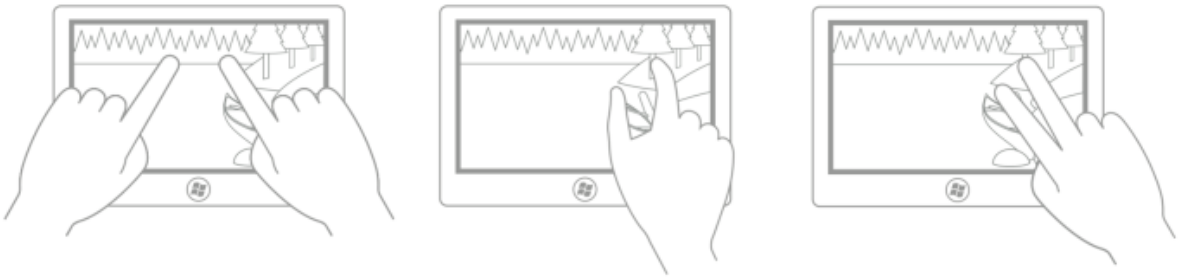
回転は Windows 8 の Windows ストア アプリで使われるタッチ操作に最適な手法であり、ユーザーがオブジェクトを回転 (時計回りまたは反時計回り) できるようにします。

入力デバイスに応じて回転操作は次のように実行されます。

- マウスまたはアクティブなペン/スタイラスを使って、選んだオブジェクトの回転グリッパーを移動する。
- タッチまたはパッシブなペン/スタイラスを使って、回転ジェスチャーによって任意の方向にオブジェクトを回転させる。

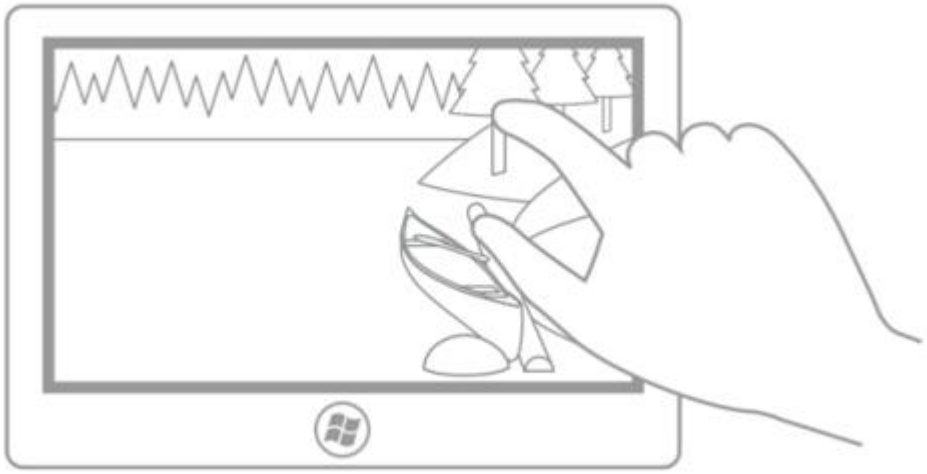
回転を使う状況

ユーザーが直接 UI 要素を回転できるように回転を使います。次の図は、サポートされる回転操作の指の配置をいくつか示しています。

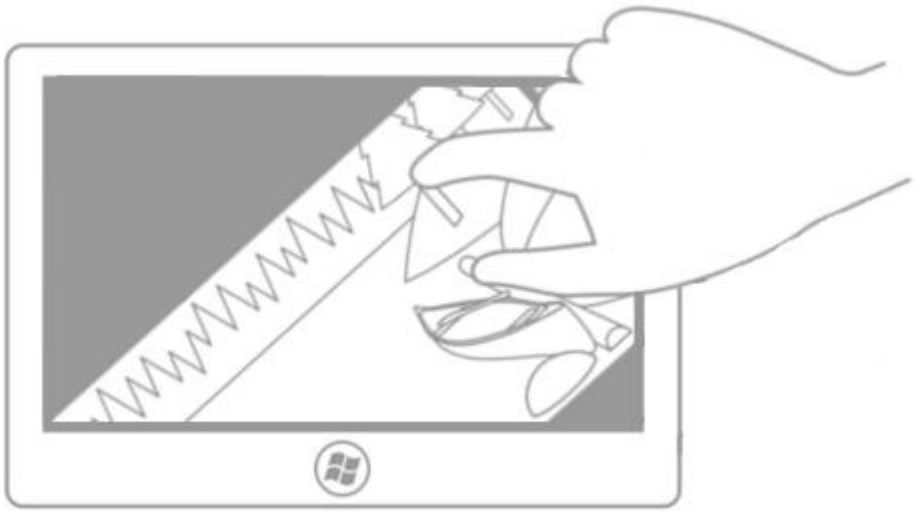


注 ユーザーが接触点とは無関係に回転の中心点を指定できる場合を除いて (例: 描画アプリやレイアウト アプリ)、直観に従い多くの場合、回転の中心点は 2 つのタッチした点のどちらかになります。以下の図では、回転の中心点がこのような制約を受けない場合に、どのようにユーザー エクスペリエンスが低下するかについて説明します。

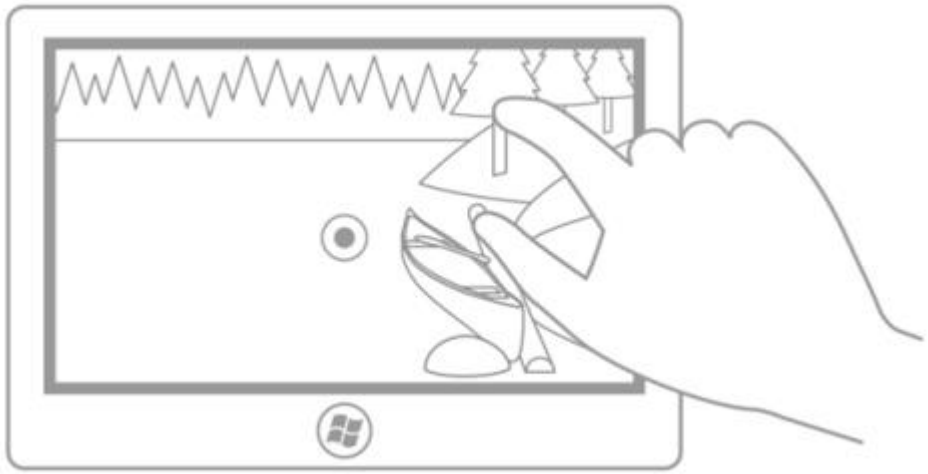
1 番目の図は、最初のタッチ ポイント (親指) と 2 番目のタッチ ポイント (人差し指) を示します。人差し指は木に、親指は丸太にタッチしています。



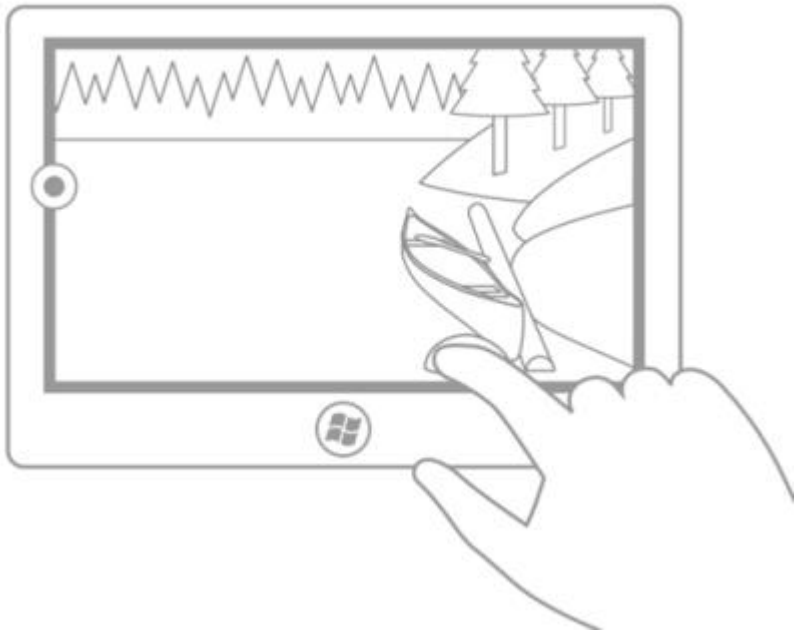
2 番目の図では、最初のタッチ ポイント (親指) の周りで回転が行われています。回転の後で、人差し指は相変わらず木の幹にタッチし、親指は相変わらず丸太 (回転の中心点) にタッチしています。



3 番目の図では、回転の中心がアプリによって絵の中心点に定義されています (またはユーザーによって設定されています)。回転の後で、絵が指の 1 つの周りで回転しなかったために、直接操作の画像が失われます (ユーザーがこの設定を選んだ場合を除きます)。



最後の図では、回転の中心がアプリによって絵の左端の中央の点に定義されています (またはユーザーによって設定されています)。この場合も、ユーザーがこの設定を選んだ場合を除いて、直接操作の画像が失われます。



Windows 8 では、自由、制約付き、複合の 3 種類の回転をサポートします。

種類	説明
自由回転	自由回転では、ユーザーはコンテンツを 360°の任意の位置に自由に回転できます。ユーザーがオブジェクトを離すと、オブジェクトは選んだ位置にとどまります。自由回転は、Microsoft PowerPoint、Word、Visio、ペイントと Adobe Photoshop、Illustrator、Flash などの描画アプリやレイアウト アプリで便利です。
制約付き回転	制約付き回転は、操作中は自由回転をサポートしますが、離したときに 90°単位のスナップ位置が強制されます (0、90、180、270)。ユーザーがオブジェクトを離すと、オブジェクトは自動的に最も近いスナップ位置まで回転します。 制約付き回転は回転の最も一般的な方法で、コンテンツのスクロールと同じように機能します。スナップ位置があることで、ユーザーは操作が正確でなくても目標の位置に到達できます。制約付きの回転は Web ブラウザーやフォト アルバムのようなアプリで便利です。
複合回転	複合回転は自由回転をサポートしますが、(パンにおけるレールのように) 90°単位のスナップ位置のゾーンでは制約付き回転によって強制されます。ユーザーが各 90°のゾーンの外でオブジェクトを離した場合にはオブジェクトはその位置にとどまりますが、それ以外の場合にはオブジェクトは自動的にスナップ位置まで回転します。 **注** ユーザー インターフェイスのレールは、ターゲットの周辺の領域において、特定の値または位置に向けて動きが制約され選択に影響を与える機能です。

テキストと画像の選択のガイドライン

このトピックでは、テキスト、画像、コントロールを選んだり操作したりするための新しい Windows UI について説明します。また、Windows ストア アプリでこの新しい選択と操作のメカニズムを使うときに考慮する必要のあるユーザー エクスペリエンスに関するガイドラインを示します。

推奨と非推奨

- 独自のグリッパー UI を実装する場合は、フォント グリフを使います。グリッパーは、システム全体で利用できる 2 つの Segoe UI フォントを組み合わせたものです。フォント リソースを使うと、さまざまな dpi におけるレンダリングの問題が軽減され、さまざまな UI 表示スケール プラトーに対応できます。独自のグリッパーを実装する場合は、どのグリッパーにも次の UI の特性を持たせてください。
  - 円の図形
  - 背景に隠れず表示される
  - 一貫したサイズ
- グリッパー UI が収まるように、選択可能なコンテンツの周囲に余白を設けます。パンとスクロールに対応していない領域でテキストを選択できる場合は、テキスト領域の左右に 1/2 のグリッパー余白を設け、テキスト領域の上下に 1 つのグリッパーの高さを設けます (次の図を参照)。こうすることで、グリッパー UI 全体がユーザーに表示されるため、エッジ (端)に基づく他の UI が誤って操作されるのを防ぐことができます。

- 対話操作中はグリッパー UI を非表示にします。対話操作中にグリッパーによって領域がふさがれないようにします。 これは、グリッパーが指で完全に隠れない場

合や、テキスト選択グリッパーが複数存在する場合に有効です。子ウィンドウを表示しているときは、ビジュアルなアーティファクトを取り除きます。

- コントロール、ラベル、画像、独自のコンテンツなどの UI 要素は選択できないようにします。通常、Windows アプリケーションでは、特定のコントロール内でのみ選択できます。ボタン、ラベル、ロゴなどのコントロールは選択できません。JavaScript を使った Windows ストア アプリでは、選択を無効にする必要があります。選択がアプリにとって問題になるかどうかを評価し、問題になる場合は、選択を禁止する UI 領域を特定します。

その他の使い方のガイダンス

テキストの選択と操作は、タッチ操作で導入されたユーザー エクスペリエンスの問題の影響を特に受けやすくなっています。マウス、ペン/スタイラス、キーボード入力は非常に細かく制御されます。1 回のマウスのクリックまたはペン/スタイラスの接触は 1 ピクセルにマッピングされ、キーは押されるか、放されます。 タッチ入力は細かく制御されません。指先の表面全体を、画面上の特定の x-y の位置にマッピングしてテキスト キャレットを正確に配置することは困難です。

考慮実行と推奨事項

Windows 8 の言語フレームワークによって公開されるビルトイン コントロールを利用して、選択や操作の動作など、完全なプラットフォームのユーザー操作エクスペリエンスを実現するアプリを作成してください。ビルトイン コントロールの対話操作の機能は、大部分の Windows ストア アプリにとって十分なものです。

標準の Windows 8 テキスト コントロールを使う場合、このトピックで説明した選択の動作と視覚効果はカスタマイズできません。

テキスト選択

アプリにテキストの選択をサポートするカスタム UI を実装する必要がある場合は、ここで説明する Windows 8 の選択動作に従うことをお勧めします。

編集可能なコンテンツと編集不可のコンテンツ

タッチ操作の精度の低いターゲット設定動作を受け入れるために、選択と操作のユーザーエクスペリエンスのビジュアル効果 (選択が調整可能でかつ操作ターゲットを特定する "グリッパー" など) が Windows 8 用に完全に再設計されています。

タッチでは、選択操作は主に挿入カーソルの設定や単語の選択を行うタップ、選択範囲の変更を行うスライドなどのジェスチャーを通じて実行されます。他の Windows 8 タッチ操作と同様に、時間制限のある対話操作は情報 UI を表示するための長押しジェスチャーに制限されます。詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください。

Windows 8 でマウス、ペン/スタイラス、キーボードを使う場合は、選択の動作に変わりはありません。

Windows 8 では、選択操作のために "編集可能" と "編集不可" の 2 つの状態が認識され、その状態に合わせて選択 UI、フィードバック、機能が調整されます。

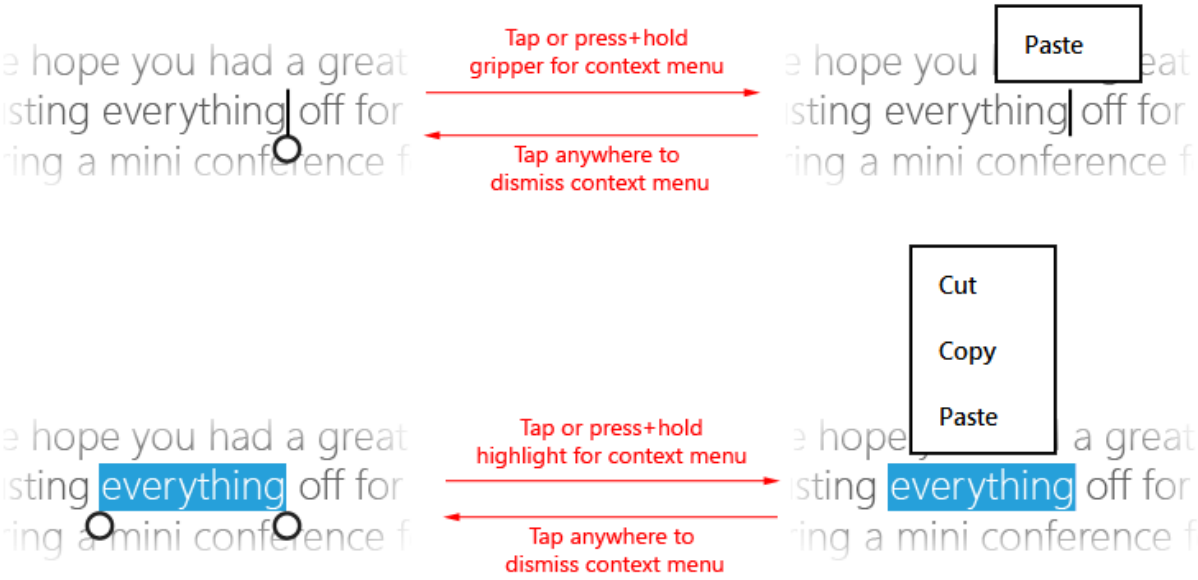
編集可能なコンテンツ

単語内の左半分をタップすると、単語のすぐ左にカーソルが配置されます。単語内の右半分をタップすると、単語のすぐ右にカーソルが配置されます。

次の図は、単語の先頭または終わりの近くでタップして、グリッパーを持つ最初の挿入カーソルを配置する方法を示しています。

次の図は、グリッパーをドラッグして選択範囲を調整する方法を示しています。

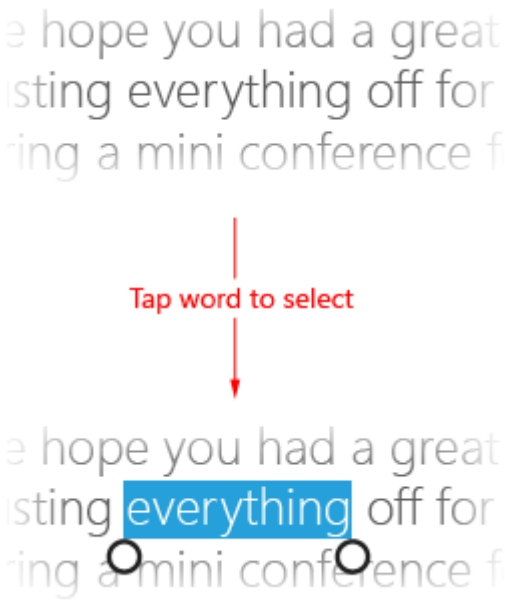
次の図は、選択範囲内またはグリッパー上でタップしてコンテキスト メニューを呼び出す方法を示しています (長押しを使うこともできます)。



注 これらの対話的操作は、綴りに間違いのある単語の場合は若干異なります。綴りに誤りがあるとしてマークされている単語をタップすると、単語全体が強調表示されて、スペル候補のコンテキスト メニューが呼び出されます。

編集不可のコンテンツ

次の図は、単語内でタップして単語を選ぶ方法を示しています (最初の選択にスペースは含まれていません)。



編集可能なテキストと同じ手順に従って、選択範囲を調整し、コンテキスト メニューを表示します。

オブジェクトの操作

Windows ストア アプリでカスタム オブジェクト操作を実装する場合は、できる限り、テキストの選択と同じ (類似する) グリッパー リソースを使います。そうすれば、プラットフォーム間で操作エクスペリエンスの一貫性が保たれます。

たとえば、次の図に示すように、グリッパーは、サイズ変更とトリミングをサポートする画像処理アプリや、調節可能なプログレス バーを備えたメディア プレーヤー アプリでも利用できます。

調節可能なプログレス バーを備えたメディア プレーヤーです。

トリミング グリッパーが表示された画像エディターです。

音声機能の設計ガイドライン(Windows Phone)

Windows Phone では、ユーザーは音声でアプリを操作できます。アプリに統合できる音声コンポーネントには、音声コマンド、音声認識、音声合成(TTS: text-to-speech) の 3 つがあります。

音声機能が適切に設計され、効果的に実装されていると、ユーザーが Windows Phone アプリを楽しく確実に操作できる手段になります。これによって、タッチ、タップ、ジェスチャによる操作を補完することも、場合によっては置き換えることもできます。

音声操作の設計

音声対応アプリのコーディングを開始する前に、ユーザー エクスペリエンスとフローを予測し、定義しておくことをお勧めします。

音声によるアプリへのアクセス

アプリには音声コマンドを統合できます。これによりユーザーは、アプリの外部からアプリにディープ リンクすることができます。たとえば、アプリ内で最もよく使われるセクションにアクセスする音声コマンドや、重要なタスクを実行する音声コマンドを追加できます。

アプリ内での音声操作

ユーザーは音声認識を使うことにより、アプリの内部から音声入力することもタスクを実行することもできます。また、アプリ内で音声合成 (TTS) を使うことにより、マイクを通じてユーザーにテキストを読み上げることもできます。

ユーザーによって (おそらく音声コマンドを使って) アプリが開かれた後の音声操作を定義するには、次の点を検討します。

- 音声を使ってどのような操作またはアプリ動作をユーザーが開始できるようにするか。たとえば、ページ間の移動、コマンドの開始、データ (メモやメッセージなど) の入力などが考えられます。
- それぞれの操作や動作を開始するために、ユーザーはどのような語句で音声入力すると考えられるか。
- アプリに話しかける適切なタイミングと内容をユーザーがどのように知るか。

- タッチより音声を使った方が速くなる可能性があるのは、どのような処理またはタスクか。たとえば、多数のオプションが含まれる一覧を参照する場合や、複数のメニュー レベル間またはページ間を移動する場合が考えられます。
- ネットワーク接続がないときに、アプリで音声認識を使用するか。
- アプリでは、カスタムのボキャブラリを要するようなユーザー グループ (医学や科学など特殊な業種、ゲーマー、特定の地域など) を対象にしているか。

音声コマンドまたはアプリ内音声認識による音声操作エクスペリエンスの組み込みを決定した場合は、次の作業を行うことができます。

- ユーザーがアプリで実行できる操作のリスト作成。
- 各操作とコマンドのマッピング。
- 各コマンドを有効化するためにユーザーが音声入力できる 1 つ以上の語句の割り当て。
- ユーザーとアプリがやり取りするダイアログの作成 (アプリからもユーザーに話しかける場合)。

音声機能設計の実装

音声コマンド

音声コマンドを有効にするには、認識可能な語句のリストを定義し、音声コマンド定義 (VCD) ファイルで各語句をコマンドにマッピングする必要があります。音声コマンドに伴う音声合成の読み上げを実現するには、実行する操作を確認するためにスピーチ シンセサイザーによって読み上げられる文字列を VCD ファイルで指定します。アプリ用の VCD ファイルを作成するときに念頭に置く必要のあるヒントを次に示します。

- ユーザーに入力を求めます。
- 音声入力のオーディオ レベルの読み上げを表示します。
- ユーザーの音声と一致した語句を確認します。
- 認識に失敗した場合は、それをユーザーに通知し、もう一度 (必要であれば繰り返して) 音声入力を行うようユーザーに求めます。
- 認識結果として可能性のある複数の候補 (存在する場合) からユーザーが選べるようにします。

Windows Phone に組み込まれている認識エクスペリエンスでは、電話でのグローバルな音声コンテキストで使用されるものと同じ対話モデルが活用されているため、ユーザーは音声入力を開始するタイミングと処理が完了したタイミングを高い確率で認識できます。組み込まれている音声も、ユーザーにとってなじみのあるものです。ユーザーはエラー時にフィードバックを受け取り、複数の一致候補がある場合にはいずれかを指定することで、あいまいさを解消することができます。詳しくは、「Windows Phone 8 のプロンプト、確認、不明瞭解消の選択肢の表示」をご覧ください。

プロンプトと確認

認識を開始する UI をユーザーに提供します。このために良い方法は、アイコンのプロパティを "Microphone" に設定したアプリ バー ボタンを使うことです。

ユーザーには、アプリの現在のコンテキストに基づいてアプリに何と話しかければよいかを知らせ、音声入力が求められている語句の例を示します。ショート メッセージ ディクテーションの入力時など、どのような内容でもユーザーの音声入力を許容する場合を除き、できる限り具体的な応答を引き出せるようなプロンプトにします。たとえば、ユーザーに "今日は何をしたいですか?" とたずねた場合、多様な応答が可能になるため、考えられる応答との照合には膨大な量の文法が必要になる可能性があります。一方、"ゲームをプレイするか、音楽を聴くか、どちらにしますか?" というプロンプトは、応答として "ゲームをプレイする" または "音楽を聴く" を具体的に求めています。2 種類の応答のみに照合するための文法であれば、大量の文法を要する場合に比べて、記述が容易になるだけでなく、おそらく認識処理の精度も高くなります。

音声認識の信頼性が低い場合には、ユーザーに確認を求めます。ユーザーの意図が明らかでない場合は、通常、ユーザーが意図していない操作をアプリで開始するよりも、ユーザーに確認を求める方が、良い結果につながります。

組み込みの認識エクスペリエンスには、プロンプトのテキストをカスタマイズできる画面、予想される音声入力の例、音声入力を確認する画面が含まれています。

認識の失敗に備えた計画

認識に失敗した場合の処理を計画します。たとえば、入力がない場合、認識の質が低い場合、または語句の一部しか認識できない場合は、アプリのロジックでこれらのケースに対処する必要があります。音声入力を理解できなかった点とユーザーが再試行できる点をユーザーに伝えることを検討します。もう一度入力できるようにする必要がある場合は、必要な入力語句として別の例をユーザーに伝え、認識を再開します。連続して何度も認識に失敗する場合は、ユーザーにテキスト入力または認識操作の終了を求めることも検討します。組み込み UI による認識エクスペリエンスには、音声認識に失敗したことをユーザーに通知し、認識を試行するためにもう一度ユーザーに音声入力を求める画面が含まれています。

オーディオ入力の問題を検出し、修正を試みます。音声認識エンジンでは、音声認識の正確さにマイナスの影響を与える可能性があるオーディオ入力の問題を検出すると、イベントが生成されます。イベント引数に格納された情報を使うと、可能であれば対処できるように、問題をユーザーに通知することができます。たとえば、音声入力の声が小さすぎる場合は、もっと大きな声で話すようにユーザーに求めることができます。組み込みの音声認識エクスペリエンスを使っているかどうかに関係なく、音声認識エンジンではこのイベントが継続的に生成されます。詳しくは、「Windows Phone 8 の音声入力の問題の処理」をご覧ください。

制約

文法とは、音声入力と照合するために音声認識エンジンが使うことのできる一連の語句を定義した、一種の制約です。音声認識エンジンでは、Windows Phone に含まれている定義済みの文法を使うことも、独自に作成したカスタムの文法を使うこともできます。ここでは、使うことのできる文法の種類の概要と、SRGS 文法を作成するためのヒントを示します。文法の種類とその用途について詳しくは、「Windows Phone 8 の文法」もご覧ください。

定義済みの文法

Windows Phone では、定義済みの 2 種類の文法がサポートされています。ユーザーが特定の言語で話す可能性がある多数の語句を照合するには、定義済みのショート メッセージ ディクテーション文法を使うことを検討します。Web クエリのコンテキストに含まれる入力

を照合するには、定義済みの Web 検索文法を使うことを検討します。このような定義済みのオンライン文法は、そのまま使うことで、10 秒までの長さの音声を認識できます。開発者による作成作業は必要ありませんが、オンライン提供であるため、実行時にはネットワークへの接続が必要になります。

カスタム文法の作成

独自の文法を作成する場合、個別の短い語句を認識するには、一覧の制約が適しています。これらの語句は、アプリからネットワークへの接続がなくても、音声認識に使い、プログラムで更新することができます。

音声認識エクスペリエンスを最大限に制御するには、独自の SRGS 文法を作成します。この方法は、1 回の認識で複数の意味をキャプチャする場合に特に効果的です。SRGS 文法は、オフラインの音声認識にも使うことができます。

SRGS 文法を作成するためのヒント

文法の規模を小さくします。文法に含める照合対象の語句を少なくする方が、規模の大きな文法に多数の語句が含まれている場合よりも、認識精度が高くなる傾向があります。一般的に望ましいのは、アプリ全体に対して 1 つの文法を設定するより、アプリの特定のシナリオごとに別々の小規模な文法を設定することです。各アプリ コンテキストで適切な音声入力内容を用意してユーザーに提示し、必要に応じて文法を有効または無効にします。これにより、各認識シナリオで音声入力を照合するために、音声認識エンジンが検索する対象を少数の語句に限定できます。

文法は、人が考えたり話したりする方法の多様性を考慮し、ユーザーがさまざまな形でコマンドを音声入力できるように設計します。たとえば SRGS 文法では、次のように GARBAGE 規則を使うことができます。

- 文法で定義されていない音声入力を照合します。これにより、"お願い"、"それと"、"ええと"、"多分" など、アプリにとって意味を持たない語句を含めて話しても、文法で明示的に定義した、アプリにとって重要な語句は正しく認識されます。

- GARBAGE 規則を一致候補リスト内の項目として追加することで、文法に定義されていない語句が誤って認識される可能性を引き下げることができます。また、音声認識エンジンで、想定外の音声入力が一致候補リスト内の GARBAGE 規則に照合された場合は、GARBAGE との一致によって返された省略記号 (…) を認識結果から検出し、音声入力をもう一度行うようにユーザーに求めることができます。ただし、一致候補リストに GARBAGE 規則を含めると、文法に定義されている語句に一致する音声入力が、誤って拒否される可能性が高くなることもあります。

GARBAGE 規則は慎重に使用し、文法が意図したとおりに動作するか試してください。詳しくは、「ruleref 要素」をご覧ください。

音声入力の認識率を高めるには、sapi:subset 要素を使用してみます。sapi:subset 要素は、有効な文法に対してユーザーの音声入力を照合しやすくするための、SRGS 仕様に対する Microsoft の拡張機能です。sapi:subset を使って定義した語句については、その語句のサブセットしか音声入力されていなくても、音声認識エンジンで照合できます。定義した語句のサブセットは、4 とおりの方法での照合に使うことができます。

音節が 1 つしかない語句は、文法に定義しないようにしてください。必要以上に長い語句の定義は避ける必要がありますが、音節が 2 つ以上ある語句の方が、正確に認識されやすくなります。

音声認識エンジンでの混乱を避けるために、一致候補リストを定義する際には、同じように聞こえる語句を使わないようにしてください。たとえば、"hello"、"bellow"、"fellow" など、響きが似ている語句を一致候補リストに含めると、認識精度が低くなる可能性があります。

文法の編集は、音声認識の準備を行ううえで必要なステップです。

カスタムの発音

特殊なボキャブラリ用に、カスタムの発音を定義することを検討します。一般的ではない単語や架空の単語、または普通とは異なる発音の単語がアプリに含まれる場合は、カスタムの発音を定義することで、認識性能が高まる可能性があります。音声認識エンジンは、エンジンの辞書に定義されていない単語についても発音を随時生成するように設計されていますが、カスタムの発音を定義しておくと、音声認識と音声合成 (TTS) の精度が高まる可能性が

あります。使用頻度の低い単語については、カスタムの発音を SRGS 文法にインラインで作成できます。詳しくは、「token 要素」をご覧ください。使用頻度の高い語句については、発音辞書のドキュメントを別途作成することもできます。詳しくは、辞書と音標文字に関するドキュメントをご覧ください。

音声認識の精度をテストする

音声認識精度のテストと、アプリでの音声認識をサポートするために用意したカスタム UI の有効性のテストは、可能であればアプリのターゲット ユーザーのグループに対して行います。アプリでの音声機能の設計と実装の有効性を判断するために最も良い方法は、アプリの音声認識精度をターゲット ユーザーに対してテストすることです。たとえば、良好な認識結果を得られない場合は、実装されている音声認識機能で、ユーザーによる音声入力の内容を想定できていない可能性があります。このような問題に対する解決策の 1 つとして、ユーザーが口にすると思われる語句を認識できるように文法を変更する方法がありますが、別の解決策としてはアプリを変更し、音声による対話が発生する前に、適切な音声入力内容をユーザーに知らせる方法もあります。テストの結果は、文法の質やアプリの音声認識フローの質を向上させる方法を見つけて、それぞれの有効性を高めるのに役立ちます。

音声合成

音声合成(TTS: text-to-speech) では、開発者が用意するテキストまたは Speech Synthesis Markup Language (SSML) の XML マークアップから音声出力が生成されます。アプリに音声合成 (TTS) を実装するための推奨事項を次に示します。

- ていねいな言葉使いで音声入力を促すようなプロンプトを設計します。
- 音声合成 (TTS) を使って、ユーザーに対して大量のテキストを読み返すかどうかを検討します。たとえば、1 つのテキスト メッセージであれば、ユーザーが TTS による読み返しを最後まで聞く可能性が高くなりますが、記憶するのが困難な長い検索結果リストを読み返した場合は、途中でユーザーが集中できなくなり混乱を招く可能性があります。
- 音声合成 (TTS) による読み上げ (特に、長くなる場合) については、中断するオプションをユーザーに提示します。

- 音声合成 (TTS) の声として、男性の声か女性の声をユーザーが選べるようにすることを検討します。Windows Phone のすべての言語では、サポート対象のロケールごとに男性と女性の声が用意されています。
- アプリを公開する前に、音声合成 (TTS) の読み上げを聞いてみます。音声シンセサイザーは明瞭かつ自然な話し方で語句を読み上げようとしますが、明瞭さまたは自然さの点で問題が生じる場合があります。
  - 明瞭さは最も重要なポイントであるため、音声合成 (TTS) で読み上げられた語句をネイティブ スピーカーが理解できるかどうかという点が反映されています。明瞭さの問題は、語句がまれなパターンで連続している場合や、数値や句読点が含まれる場合に、発生する可能性があります。
  - 自然さは必要とされるポイントであり、読み上げの韻律や抑揚がネイティブ スピーカーによる発声と異なると、問題になる場合があります。どちらの問題も、シンセサイザーへの入力にプレーンテキストではなく SSML を使用することで対処できます。SSML について詳しくは、SSML による合成音声の制御に関するページと、Speech Synthesis Markup Language (SSML) のリファレンスをご覧ください。

ターゲット設定のガイドライン

Windows のタッチ補正では、タッチ デジタイザーで検出されるそれぞれの指が接触する領域全体を使います。デジタイザーから伝えられる、より広く複雑なこの入力データのセットを使うと、ユーザーが意図した (または意図した可能性が高い) ターゲットをより正確に特定できます。

このトピックでは、タッチ補正のための接触形状の使用について説明し、Windows ランタイム アプリでのターゲット設定のベスト プラクティスを紹介します。

サイズと表示スケール

画面サイズとピクセル密度が変わっても一貫性が維持されるように、ターゲット サイズはすべて物理単位 (ミリメートル) で表されます。物理単位は、次の式でピクセルに変換できます。

ピクセル数 = ピクセル密度 × サイズ

次の例ではこの式を使って、135 ppi (pixel per inch) のディスプレイと 1x の表示スケール プラトーでの 9 mm ターゲットのピクセル サイズを計算します。

ピクセル数 = 135 ppi × 9 mm

ピクセル数 = 135 ppi × (0.03937 インチ/mm × 9 mm)

ピクセル数 = 135 ppi × 0.35433 インチ

ピクセル数 = 48 ピクセル

この結果は、システムで定義されている各表示スケール プラトーに従って調整する必要があります。

しきい値

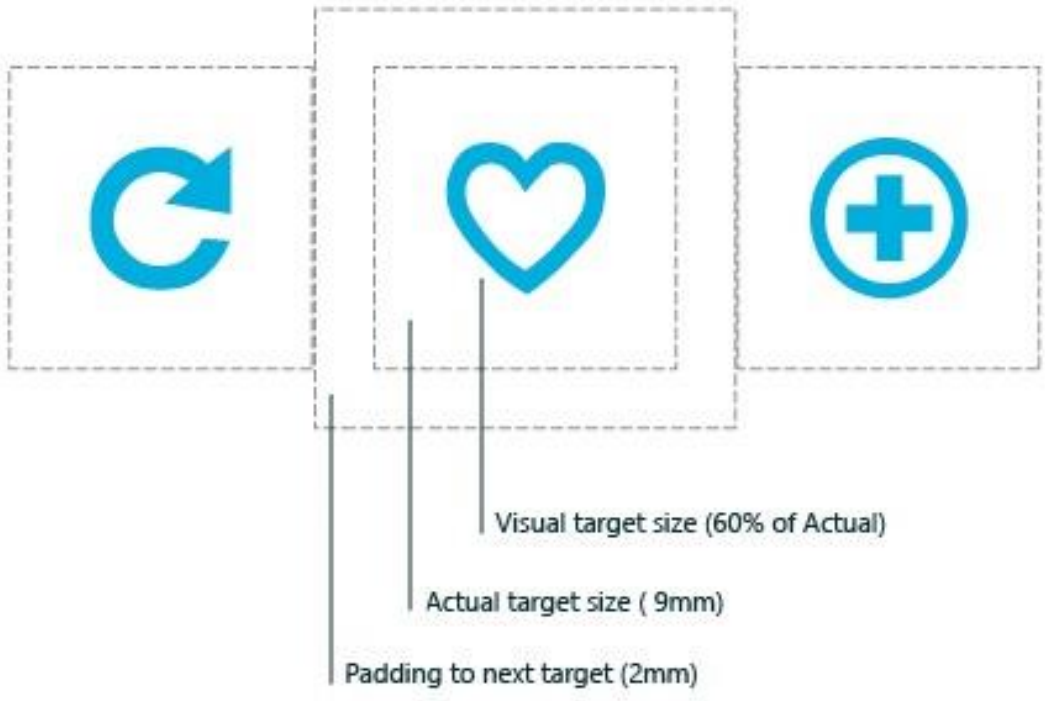
操作の結果を判断するために距離と時間のしきい値を使うことがあります。

たとえば、タッチ ダウンが検出されたとき、オブジェクトがタッチ ダウン ポイントから 2.7 mm 未満の範囲でドラッグされて、タッチ ダウンから 0.1 秒以内に指が上げられた場合は、タップが登録されます。この 2.7 mm のしきい値を超えて指を動かすと、オブジェクトはドラッグされ、選択または移動されます (詳しくは、「クロススライドのガイドライン」をご覧ください)。アプリによっては、0.1 秒より長く押し続けると自己説明操作が実行されることもあります (詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください)。

ターゲット サイズ

一般に、タッチ ターゲットのサイズを一辺 9 mm 以上の正方形に設定します (1.0x のスケール プラトーで 135 ppi のディスプレイで 48x48 ピクセル)。一辺 7 mm 未満の正方形であるタッチ ターゲットを使わないようにしてください。

次の図は、ターゲット サイズが一般には外観上のターゲット、実際のターゲット サイズ、実際のターゲットと他の指定可能なターゲットの間の余白の組み合わせで決まることを示しています。



タッチ ターゲットのすべてのコンポーネントの最小サイズと推奨サイズを次の表に示します。

ターゲット コンポーネント	最小サイズ	推奨サイズ
余白	2 mm	該当なし
外観上のターゲット サイズ	実際のサイズの 60% 未満	実際のサイズの 90 から 100% ほとんどのユーザーは一辺 4.2 mm の正方形 (7 mm の推奨の最小ターゲットのサイズの 60%) よりも小さい場合、外観上のターゲットがタッチ可能であると実感しません。
実際のターゲット サイズ	7 x 7 mm の正方形	一辺 9 mm 以上の正方形 (48 x 48 ピクセル @ 1x)
全体的のターゲット サイズ	11 x 11 mm (約 60 ピクセル: 3 個の 20 ピクセル グリッド @ 1x)	13.5 x 13.5 mm (72 x 72 ピクセル @ 1x) これは、実際のターゲットと余白を組み合わせたサイズをそれぞれの最小サイズより大きくする必要があることを意味します。

表に示したターゲット サイズの推奨サイズは、個々のシナリオの必要に応じて調整できます。この推奨サイズの設定では、次の点が考慮されています。

- タッチの頻度: 繰り返しタッチされたり、頻繁にタッチされたりするターゲットは、最小サイズより大きくするようにしてください。
- エラー防止: 誤ってタッチすると重大な結果をもたらすターゲットは、大きな余白を取り、コンテンツ領域の端から離して配置する必要があります。特に当てはまるのは頻繁にタッチされるターゲットです。
- コンテンツ領域での位置
- フォーム ファクターと画面サイズ
- 指の位置
- タッチのビジュアル エフェクト

- ハードウェアとタッチ デジタイザー

ターゲット設定支援

Windows では、ここで示した最小サイズや推奨する余白サイズを適用できない状況に対応するためのターゲット設定支援機能を提供しています。対象となるのは、Web ページ上のハイパーリンク、カレンダー コントロール、ドロップダウン リストとコンボ ボックス、テキスト選択などです。

このようにターゲット設定プラットフォームを強化し、ユーザー インターフェイスの動作をビジュアルなフィードバック (不明瞭解消 UI) と連携させることで、ユーザーがより正確に、また安心して操作できるようになります。 詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください。

タッチ可能な要素を推奨の最小ターゲット サイズより小さくする必要がある場合は、次のテクニックを使ってターゲット設定で発生する問題を最小化できます。

テザー

テザーとは、入力接点がオブジェクトに直接触れていなくても、ユーザーがそのオブジェクトにつながっているか操作していることをユーザーに示すために使われる視覚的な合図 (接点からオブジェクトの境界の四角形までを結ぶコネクタ) です。テザーは次の場合に使われます。

- タッチによる接触がオブジェクトまでの近接しきい値の範囲内で最初に検出され、そのオブジェクトがそのタッチのターゲットとして最も可能性が高いと特定された場合。
- タッチによる接触がオブジェクトから離れたが、その接触が近接しきい値内にとどまっている場合。

この機能は、JavaScript を使った Windows ストア アプリの開発者には公開されていません。

スクラブ

スクラブとは、ターゲットのフィールド内をタッチし、そのまま指を持ち上げずに目的のターゲットまでスライドさせてそのターゲットを選ぶことを意味します。 この操作は "指を離すことによるアクティブ化" とも呼ばれます。この場合、アクティブ化されるオブジェクトは、指が画面から離れたときに最後にタッチされたオブジェクトです。

スクラブ操作を設計するときは、次のガイドラインに従ってください。

- スクラブは、不明瞭解消 UI と併用します。詳しくは、「ビジュアルなフィードバックのガイドライン」をご覧ください。
- スクラブ対象のタッチ ターゲットの推奨最小サイズは、20 ピクセル (3.75 mm @ 1x サイズ) です。
- スクラブは、Web ページなどのパン対応サーフェイスで実行されたときに優先されます。
- スクラブ対象ターゲットは互いに近づける必要があります。
- ユーザーがドラッグによってスクラブ対象ターゲットから指を離すと、操作は取り消されます。
- ターゲットによって実行される操作が破棄的でなければ (カレンダーでの日付の切り替えなど)、スクラブ対象ターゲットに対してテザーが指定されます。
- テザーは、水平、垂直のいずれかの方向で指定されます。

タッチ操作のガイドライン

タッチ用に最適化される一方で、さまざまな入力デバイスで一貫した機能を提供する、直観的で独特なユーザー操作エクスペリエンスを備えた Windows ストア アプリを作成します。

Windows の魅力的な操作機能を活用できます。次のガイドラインは、そのために役立ちます。

推奨と非推奨

- 期待される主な入力方法としてタッチ操作を使うアプリを設計します。
- あらゆる種類 (タッチ、ペン、スタイラス、マウスなど) の操作に対するビジュアルなフィードバックを提供します。
- タッチ ターゲットのサイズ、接触形状、スクラブ、揺らす操作を調整してターゲット設定を最適化します。
- スナップ位置と方向 "レール" を使って精度を最適化します。
- 密集した UI 項目のタッチの精度を高めるためにヒントとハンドルを用意します。
- できるだけ時間制限のある操作を使わないようにします (適切な使用例: 長押し)。
- できる限り、操作の区別に使われた数の指は使わないようにします。

その他の使い方のガイダンス

まず、タッチがユーザーの主な入力方法になるという想定でアプリを設計します。プラットフォーム コントロールを使う場合は、タッチパッド、マウス、ペン/スタイラスをサポートするために追加のプログラミングを行う必要はありません。Windows 8 では、それらが標準で提供されます。

ただし、タッチ用に最適化された UI が従来の UI よりも常に優れているとは限らないことに留意してください。どちらの UI にも、テクノロジとアプリに固有の長所と短所があります。タッチ操作主体の UI に移行する際に、タッチ (タッチパッドを含む)、ペン/スタイラス、マウス、キーボードの各入力の主な違いを理解することが重要です。Windows 8 のタッチは単に機能をエミュレートするだけではないので、使い慣れた入力デバイスのプロパティと動作に変化がないとは考えないでください。

このガイドラインに目を通すと、タッチ入力では異なる UI 設計方法が必要になるとわかります。

タッチ操作の要件の比較

次の表に、タッチ操作に最適な Windows ストア アプリの設計時に考慮する必要がある、入力デバイス間の違いをいくつか示します。

要因	タッチ操作	マウス、キーボード、ペン/スタイラス操作	タッチパッド
正確性	指先が接触する領域は単一の XY 座標よりも広いので、意図していないコマンドがアクティブ化される可能性が高くなります。	マウスとペン/スタイラスを使うと正確な XY 座標を指定できます。	マウスと同じです。
	接触する領域の形は移動を通じて変化します。	マウスの移動とペン/スタイラスのストロークによって正確な XY 座標を指定できます。キーボード フォーカスは明示的です。	マウスと同じです。
	ターゲット設定に役立つマウス カーソルはありません。	マウス カーソル、ペン/スタイラス カーソル、キーボード フォーカスはすべてターゲット設定に役立ちます。	マウスと同じです。
人体構造	1 本または複数の指で直線移動を行うのは困難なので、指先の動きは正確さに欠けます。これは、手関節が曲がることや動きに関係する関節の数が原因です。	マウスまたはペン/スタイラスを制御する手の物理的な移動距離は、画面上のカーソルの移動距離よりも短いので、マウスまたはペン/スタイラスで直線移動を行う方が簡単です。	マウスと同じです。
	ディスプレイ デバイスのタッチ画面には、指の位置とユーザーのデバイスの持ち方が原因で届きにくくなる領域もあります。	マウスとペン/スタイラスは画面のどの部分にも届き、キーボードではタブ オーダーによってどのコントロールにもアクセスできます。	指の位置と持ち方が問題になることがあります。

	オブジェクトは、1 本以上の指先またはユーザーの手で隠れる場合があります。これをオクルージョンと呼びます。	間接的な入力デバイスでは、オクルージョンは発生しません。	マウスと同じです。
オブジェクトの状態	タッチでは、ディスプレイ デバイスのタッチ画面がタッチされているか (オン) タッチされていないか (オフ) の 2 状態モデルが使われます。追加のビジュアルなフィードバックをトリガーできるホバー状態はありません。	マウス、ペン/スタイラス、キーボードはすべて、離した状態 (オフ)、押した状態 (オン)、ホバー状態 (フォーカス) の 3 状態モデルを公開します。ホバーすると、UI 要素に関連付けられたツールチップを調べて参考にすることができます。ホバーまたはフォーカスを合わせたときの効果によって操作可能なオブジェクトがわかり、ターゲット設定にも役立ちます。	マウスと同じです。
豊富な操作	タッチ画面において複数の入力ポイント (指先) で操作できるマルチタッチをサポートします。	単一の入力ポイントをサポートします。	タッチと同じです。
	タップ、ドラッグ、スライド、ピンチ、回転などのジェスチャーによるオブジェクトの直接操作をサポートします。	マウス、ペン/スタイラス、キーボードは間接的な入力デバイスなので、直接操作はサポートされません。	マウスと同じです。

注　間接的な入力には、25 年以上の改良を経ているという利点があります。ホバーすると表示されるツールチップなどの機能は、タッチパッド、マウス、ペン/スタイラス、キーボード入力での UI の操作を解決するために特別に設計されています。Windows タッチでは、このような UI 機能は、他のデバイスのユーザー エクスペリエンスを損なうことなくタッチ入力の充実したエクスペリエンスの長所を活かす方法で設計されています。

タッチのフィードバックの使用

アプリの対話的操作中にビジュアルなフィードバックが適切に表示されると、その対話的操作がアプリと Windows 8 の両方でどのように解釈されるかに関する認識、学習、適応に役立ちます。ビジュアルなフィードバックの用途は、対話的操作の成功の表示、システム状態の中継、コントロール感の向上、エラーの低減、システムと入力デバイスに関するユーザーの理解の支援、対話的操作の促進などです。

位置に基づく正確性が求められる操作をタッチ入力で行う場合は、ビジュアルなフィードバックが重要です。タッチ入力が検出された場所に必ずフィードバックを表示して、アプリとそのコントロールで定義されたカスタム ターゲット設定規則をユーザーが把握できるようにします。

イマーシブな操作性の実現

次の方法を使って、Windows ストア アプリのイマーシブな操作性を高めます。

ターゲット設定

ターゲット設定は、次の要素によって最適化します。

- タッチ ターゲットのサイズ
  明確なサイズのガイドラインによって、ターゲット設定しやすいオブジェクトとコントロールが含まれる快適な UI を備えたアプリになるようにします。
- 接触形状
  指が接触する領域全体によって、意図された可能性が最も高いターゲット オブジェクトを特定します。
- スクラブ
  グループ内の項目 (ラジオ ボタンなど) 間で指をドラッグすると、それらの項目を簡単にターゲット設定し直すことができます。指を離すと現在の項目がアクティブ化されます。
- 揺らす
  密集した複数の項目 (ハイパーリンクなど) を指で押してスライドせずに前後に揺らすと、それらの項目を簡単にターゲット設定し直すことができます。 オクルージ

ョンが原因で、現在の項目はツールチップまたはステータス バーで特定され、指を離すとアクティブ化されます。

正確性

以下を使って、対話的操作が雑な場合に備えて設計します。

- コンテンツの操作時に目的の位置で簡単に停止できるようにするスナップ位置。
- 手をわずかに曲げて動かした場合でも垂直方向または水平方向のパンを実行できる方向 "レール"。詳しくは、「パンのガイドライン」をご覧ください。

オクルージョン

指と手のオクルージョンは、次の要素によって回避します。

- UI のサイズと配置
  UI 要素を十分に大きくして、指先が接触する領域で完全にふさぐことができないようにします。
  メニューとフライアウトは、できる限り接触する領域の上に配置します。
- ツールチップ
  指がオブジェクトに接触し続けているときは、ツールチップを表示します。オブジェクトの機能について説明する場合に便利です。ツールチップが呼び出されないようにするには、ユーザーは指先をオブジェクトの外にドラッグします。
  小さいオブジェクトの場合は、ツールチップをずらして、指先が接触する領域でふさがれないようにします。これはターゲット設定に役立ちます。
- 正確さを確保するためのハンドル
  正確さが要求される場合 (テキスト選択など)、正確さを向上させるためにオフセットされる選択ハンドルを用意します。詳しくは、「テキストと画像の選択のガイドライン」をご覧ください。

タイミング

直接操作を行うために、時間制限のあるモードの変更を避けます。直接操作は、オブジェクトに対するリアルタイムで物理的な直接処理をシミュレートします。オブジェクトは指の動きに合わせて反応します。

一方、時間制限のある操作は、タッチ操作の後に発生します。通常、時間制限のある操作では、時間、距離、速度などの見えないしきい値に基づいて実行するコマンドが決定されます。システムで操作が実行されるまで、時間制限のある操作ではビジュアルなフィードバックは返されません。

直接操作には、時間制限のある対話操作と比べて、いくつかの利点があります。

- 対話操作中にビジュアルなフィードバックがすばやく返されるので、ユーザーはより集中して、すべてを制御しているという安心感を持って操作できます。
- 直接操作は元に戻すことができるので、安心してシステムを使うことができます。ユーザーは、論理的かつ直観的な方法で操作を簡単にさかのぼることができます。
- オブジェクトに直接影響して実際の対話操作を模倣する操作は、より直観的で見つけやすく覚えやすい対話操作です。わかりにくい抽象的な対話操作には依存しません。
- 時間制限のある対話操作は、任意の見えないしきい値に達する必要があるので、実行が難しい場合があります。

また、次の点を考慮することを強くお勧めします。

- 操作は、使う指の数で区別しないでください。
- 複合操作をサポートしてください。たとえば、ピンチによるズームを行いながら指をドラッグしてパンできるようにします。
- 対話操作を時間で区別しないでください。実行にかかる時間に関係なく、同じ対話操作を行うと同じ結果が得られるようにします。時間ベースのアクティブ化では、ユーザーは遅延を強いられるので、直接操作のイマーシブの特性が損なわれ、システムの応答性が低く感じられるようになります。

  注 ただし、特定の時間制限のある対話操作を使って学習や調査に役立てる場合は例外です (長押しなど)。

- 適切な説明とビジュアルな合図を使うと、高度な対話操作を非常に効果的に使用できます。

タッチ キーボードのガイドライン

タッチ キーボードを使うと、タッチをサポートしていて組み込みまたは外付けのキーボードを持たないデバイス上の Windows ストア アプリでもテキストを入力できます。 タッチ キーボードは、編集できる入力フィールドをユーザーがタップすると表示され、入力フィールドからフォーカスが移動すると消えます。タッチ キーボードはテキスト入力のみに使われ、Alt キーやファンクション キーなどのコマンド キーは用意されていません。

推奨と非推奨

- タッチ キーボードは、テキスト入力にのみ表示します。 タッチ キーボードは、ハードウェア キーボードにあるようなアクセラレータやコマンド キー (Alt キー、ファンクション キー、Windows ロゴ キーなど) の多くを備えていないことに注意してください。
- タッチ キーボードからの入力をコマンドやショートカットに定義し直さないでください。
- ユーザーがタッチ キーボードを使ってアプリを操作しないようにしてください。

タッチ キーボードとカスタム UI

これらの推奨事項は、アプリでカスタム コントロールを使う場合にのみ関係します。 Windows に用意されているテキスト入力コントロールは、既定で、タッチ キーボードを正しく操作します。C#/VB/C++ と XAML を使ってカスタム UI を作成する場合は、AutomationPeer class を使って UI オートメーションのコントロールにアクセスしてくださ

い。入力: タッチ キーボードのサンプルに関するページでは、XAML を使ったアプリのカスタム コントロールを利用してタッチ キーボードを起動する方法について説明します。

JavaScript と HTML を使っている場合、UI オートメーションにアクセスするには、カスタム コントロールの Accessible Rich Internet Applications (ARIA) プロパティを設定する必要があります。詳しくは、「UI オートメーションへの ARIA ロールのマッピング」をご覧ください。

- フォームに対する操作が行われている間はキーボードを表示します。
  テキスト入力のコンテキストでフォーカスがテキスト入力フィールドから移動したときにキーボードの表示を持続させるには、カスタム コントロールに適切な UI オートメーションの ControlType を使います。たとえば、テキスト入力シナリオの半ばでメニューを開くときに、キーボードを表示したままにするには、このメニューに ControlType Menu が必要です。C#/VB/C++ と XAML を使ったアプリについては、「AutomationControlType 列挙値」をご覧ください。JavaScript と HTML については、ARIA ロールを適切なコントロール型に設定します。
- 入力しているフィールドをユーザーが常に見られるようにします。
  タッチ キーボードを表示すると画面の半分が見えなくなります。Windows ストア アプリでは、タッチ キーボードが表示されたときに、フォーカスのある入力フィールドをビュー内にスクロールすることで UI を管理する既定のエクスペリエンスが用意されています。UI をカスタマイズする場合は、InputPane オブジェクトで公開される Showing イベントと Hiding イベントを処理して、キーボードの表示に対するアプリの反応をカスタマイズします。
- UI オートメーション プロパティを操作してタッチ キーボードを制御しない。UI オートメーション プロパティの正確さに依存する他のアクセシビリティ ツールがあります。
- テキスト入力できるカスタム コントロールの UI オートメーション プロパティを実装します。別のコントロールにフォーカスが移動したときに、コンテキストに応じてキーボードの表示を持続させるには、カスタム コントロールに次のいずれかのプロパティが必要です。
  - ボタン
  - チェック ボックス
  - コンボ ボックス

- ラジオ ボタン
- スクロール バー
- ツリー項目
- メニュー
- メニュー項目

C#/VB/C++ と XAML を使ったアプリについては、「AutomationControlType 列挙値」をご覧ください。JavaScript と HTML については、ARIA ロールを適切なコントロール型に設定します。

その他の使い方のガイダンス

Showing イベントと Hiding イベントの処理

タッチ キーボードの **Showing** イベントと **Hiding** イベントのイベント ハンドラーをアタッチする例を次に示します。

```
public class MyApplication
{
    public MyApplication()
    {
        // Grab the input pane for the main application window and attach
        // touch keyboard event handlers.
        Windows.Foundation.Application.InputPane.GetForCurrentView().Showing
            += new EventHandler(_OnInputPaneShowing);
        Windows.Foundation.Application.InputPane.GetForCurrentView().Hiding
            += new EventHandler(_OnInputPaneHiding);
    }

    private void _OnInputPaneShowing(object sender, IInputPaneVisibilityEventArgs
eventArgs)
    {
        // If the size of this window is going to be too small, the app uses
        // the Showing event to begin some element removal animations.
        if (eventArgs.OccludedRect.Top < 400)
        {
            _StartElementRemovalAnimations();

            // Don't use framework scroll- or visibility-related
            // animations that might conflict with the app's logic.
            eventArgs.EnsuredFocusedElementInView = true;
        }
    }

    private void _OnInputPaneHiding(object sender, IInputPaneVisibilityEventArgs
eventArgs)
    {
        if (_ResetToDefaultElements())
        {
            eventArgs.EnsuredFocusedElementInView = true;
        }
    }

    private void _StartElementRemovalAnimations()
    {
        // This function starts the process of removing elements
        // and starting the animation.
    }

    private void _ResetToDefaultElements()
    {
        // This function resets the window's elements to their default state.
    }
}
```

```
<SCRIPT type="text/javascript">
   Windows.UI.Immersive.InputPane.getForCurrentView().addEventListener("showing",
onInputPaneShowing);
   Windows.UI.Immersive.InputPane.getForCurrentView().addEventListener("hiding",
onInputPaneHiding);

// Handle the showing event.
function onInputPaneShowing(e)
{
   var occludedRect = e.occludedRect;

   // For this hypothetical application, the developer decided that 400 pixels is
   // the minimum height that will work for the current layout. When the
   // app gets the InputPaneShowing message, the pane is beginning to animate in.

   if (occludedRect.Top < 400)
   {
      // In this scenario, the developer decides to remove some elements (perhaps
      // a fixed navbar) and dim the screen to give focus to the text element.
      var elementsToRemove = document.getElementsByName("extraneousElements");

      // The app developer is not using default framework animation.
      _StartElementRemovalAnimations(elementsToRemove);
      _StartScreenDimAnimation();
   }

   // This developer doesn't want the framework's focused element visibility
   // code/animation to override the custom logic.
   e.ensuredFocusedElementInView = true;
}

// Handle the hiding event.
function onInputPaneHiding(e)
{
   // In this case, the Input Pane is dismissing. The developer can use
   // this message to start animations.
   if (_ExtraElementsWereRemoved())
   {
      _StartElementAdditionAnimations();
   }

   // Don't use framework scroll- or visibility-related
   // animations that might conflict with the app's logic.
   e.ensuredFocusedElementInView = true;
}
```

ビジュアルなフィードバックのガイドライン

ビジュアルなフィードバックは、Windows ストア アプリの対話操作が検出、解釈、処理されていることをユーザーに示すために使います。 ビジュアル的なフィードバックは、対話操作を促進することによってユーザーを支援します。対話操作の成功を示すことによって、ユーザーのコントロール感を向上させます。また、システム状態の中継やエラーの削減も可能になります。

推奨と非推奨

- たとえ接触時間が短くても、ビジュアルなフィードバックを返す必要があります。理由は次のとおりです。
  - タッチ スクリーンが機能していることを示すため。
  - ターゲットがタッチに対応しているか、応答できるかを示すため。
  - タッチしたところが目的のターゲットから外れているかどうかを示すため。
- すべての操作イベントに対してフィードバックをすぐに表示します。
- ユーザーの注意をそらさない、繊細かつ直感的なフィードバックを提供します。
- すべての操作において、タッチ ターゲットが指先から離れないようにします。
- パンの方向が 1 方向に制限されている場合は、スワイプ ジェスチャーを使った項目の選択を有効にします。
- タッチのビジュアル エフェクトがアプリの使用を妨げる可能性がある場合は、使わないでください。
- どうしても必要な場合以外は、フィードバックを表示しないでください。その場所でしか意味がない場合を除き、ビジュアルなフィードバックを表示せず UI をすっきりさせてください。既に表示されているテキストをさらに表示することになる場合は、ツールチップを表示しないでください。ツールチップは、項目を選択するときにテキストが途中までしか表示されていない (テキストに省略記号が付いている) 場合や、アプリの理解や使用に追加情報が必要な場合など、特定の場面でのみ使います。
- 情報提供型 UI 以外には長押しジェスチャーを使わないでください。

  重要 水平方向と垂直方向のパンが有効である場合は、長押しを選択に使うことができます。

- Windows 8 の組み込みジェスチャーのビジュアルなフィードバックの動作をカスタマイズしないでください。この動作をカスタマイズすると、ユーザー エクスペリエンスに一貫性がなくなり、混乱する可能性があります。
- パンやドラッグの操作中はビジュアルなフィードバックを返さないでください。画面上のオブジェクトの動きだけで十分です。ただし、コンテンツ領域がパンまたはスクロールしない場合は、ビジュアル エフェクトで境界条件を示します。詳しくは、「パンのガイドライン」をご覧ください。
- ターゲットとして識別されないコントロールにはフィードバックを表示しないでください。位置に基づく正確性が求められる操作をタッチ入力で行う場合は、ビジュアルなフィードバックが重要です。タッチ入力が検出されるたびにフィードバックが表示されるようにすると、ユーザーは、アプリとそのコントロールで定義されたカスタム ターゲット設定ヒューリスティックを把握できます。
- 特定の種類の入力に対するフィードバック動作を、別の種類の入力に使わないでください。たとえば、キーボード フォーカスの四角形はキーボード入力のみに使い、タッチ操作には使わないでください。

その他の使い方のガイダンス

正確性が求められるタッチ操作では、接触のビジュアル エフェクトが特に重要です。たとえば、アプリでタップの位置を正確に示し、対象から外れていたかどうか、どの程度外れていたか、合わせるにはどうすればよいかをユーザーが把握できるようにしなければなりません。

Windows ストア アプリの言語フレームワーク (JavaScript を使った Windows ストア アプリ、C++、C#、Visual Basic を使った Windows ストア アプリ) を通じて公開されるプラットフォーム コントロールを利用すると、Windows 8 のビジュアル エフェクトを標準機能として実装できます。カスタマイズされたフィードバックが必要なカスタム操作をアプリに実装する場合は、適切なフィードバックを実装し、入力デバイス間の連絡を取り、ユーザーがタスクに集中できるようにする必要があります。このことは、ビジュアルなフィードバックが重要な UI と競合したり、重要な UI を隠したりする可能性があるゲームや描画アプリでは特に重要です。

重要　組み込みジェスチャーの操作の動作を変更することはお勧めしません。

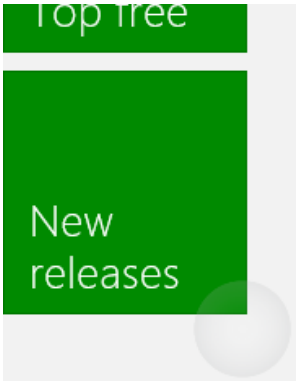
フィードバック UI

フィードバック UI は、一般に入力デバイス (タッチ、タッチパッド、マウス、ペン/スタイラス、キーボードなど) に依存します。たとえば、マウスの組み込みフィードバックには、通常はカーソルの移動と変化が伴います。一方、タッチとペンの場合は接触のビジュアル エフェクトが必要です。キーボードによる入力とナビゲーションの場合は、フォーカス用の四角形と強調表示を使います。

プラットフォーム ジェスチャーのフィードバック動作を設定するには、ShowGestureFeedback を使います。

フィードバック UI をカスタマイズする場合は、すべての入力モードをサポートした適切なフィードバックを提供してください。

Windows 8 には、次のような接触のビジュアル エフェクトが組み込まれています。



タッチのビジュアル エフェクト

マウス/タッチパッドのビジュアル エフェクト

ペンのビジュアル エフェクト

キーボードのビジュアル エフェクト

情報提供型 UI(ポップアップ)

ビジュアルなフィードバックの主な形態の 1 つに、情報提供型 UI (不明瞭解消型 UI) があります。 情報提供型 UI は、オブジェクトに関する情報を識別および表示し、その機能の説明と使用方法を示します (詳しくは、「タッチ操作の設計」をご覧ください)。さらに、必要に応じてガイダンスも提供します。

次に、Windows ストア アプリでサポートされているさまざまな情報提供型 UI の種類を示します。

- ツールチップ
- リッチ ツールチップ
- メニュー
- メッセージ ダイアログ
- フライアウト

情報提供型 UI は、指先によるオクルージョン (干渉) を克服し、アプリのタッチ操作を向上させるうえで特に有用です。長押しというこの UI 専用の組み込みジェスチャーも用意されています。

長押しは時間制限のある操作であり、一般に Windows 8 では推奨されていません。学習や調査に役立つため、この場合は時間制限のある操作を使ってもかまいません。推奨される時間は情報提供型 UI の種類によって変わります。推奨される時間のしきい値を次に示します。

情報提供型 UI の種類	**タイミング**	**アクティブ化**	**用途**
オクルージョン ヒント (スクラブ操作と小さなターゲット用)	0 ミリ秒	あり	操作についてすばやく説明するために使います。 通常はコマンドに使います。
オクルージョン ヒント (操作用)	200 ミリ秒	あり	
リッチ ヒント	2000 ミリ秒以下	なし	比較的低速で意図的な調査や学習に使います。 通常はコレクション項目に使います。
インストラクショナル	2000 ミリ秒以下	なし	
コンテキスト メニュー	2000 ミリ秒以下	なし	選択したオブジェクトに関連する限られたコマンドのセットを表示します。
フライアウト	2000 ミリ秒以下	なし	選択したオブジェクトに関連する限られたコマンドのセットを表示します。

ツールチップ

ツールチップを使ってコントロールに関する詳しい情報を表示してから、ユーザーに操作を実行するように求めます。

ツールチップ (Tooltip) は、ユーザーがコントロールまたはオブジェクト上で長押しジェスチャーを行った (またはホバー イベントが検出された) ときに自動的に表示されます。接触またはカーソルがコントロールやオブジェクトの外に移動すると消えます。ツールチップにはテキストや画像を含めることができますが、対話的な操作はできません。

小さなターゲット用のオクルージョン ツールチップ

オクルージョン ツールチップは、隠れたターゲットについて説明するために使います。Web ページ上のハイパーリンクなど、標準のタッチ ターゲット サイズよりも小さな項目をターゲット設定してアクティブ化するときに、これらのツールチップが役立ちます。

これらのツールチップは、一定の時間しきい値の経過後に情報ポップアップに変えることができます。 たとえば、オクルージョン ツールチップを使って、隠れたハイパーリンク テキストを示した後で、このツールチップを URL を含むポップアップに変えることができます。

操作とコマンド用のオクルージョン ツールチップ

これらのツールチップは、ユーザーが要素から指を離したときに発生するアクションまたはコマンドについて説明するために使います。ボタンやそれに似たコントロールをターゲット設定してアクティブ化するときに、これらのツールチップが役立ちます。

小さなターゲットのツールチップは、一定の時間しきい値の経過後にアクションのツールチップに変えることができます。 この場合、小さなターゲットのツールチップに情報を追加してアクションのツールチップにします。

リッチ ツールチップ

これらのツールチップでは、要素に関する補助的な情報を提供します。たとえば、画像の説明、途中で切れているタイトルの完全なテキストなどの、ターゲットに関する情報です。

一般的に、リッチ ツールチップには、すぐには必要でない情報や表示のタイミングが早すぎると煩わしいような情報を含めます。 時間のしきい値を長くすると、情報を得ようとするユーザーの意図が強まります。

リッチ ツールチップが表示された後にユーザーが指を離すと、オブジェクトはアクティブでなくなります。 ツールチップから情報を得たユーザーが、その項目をアクティブ化しようと思わない場合もあるためです。

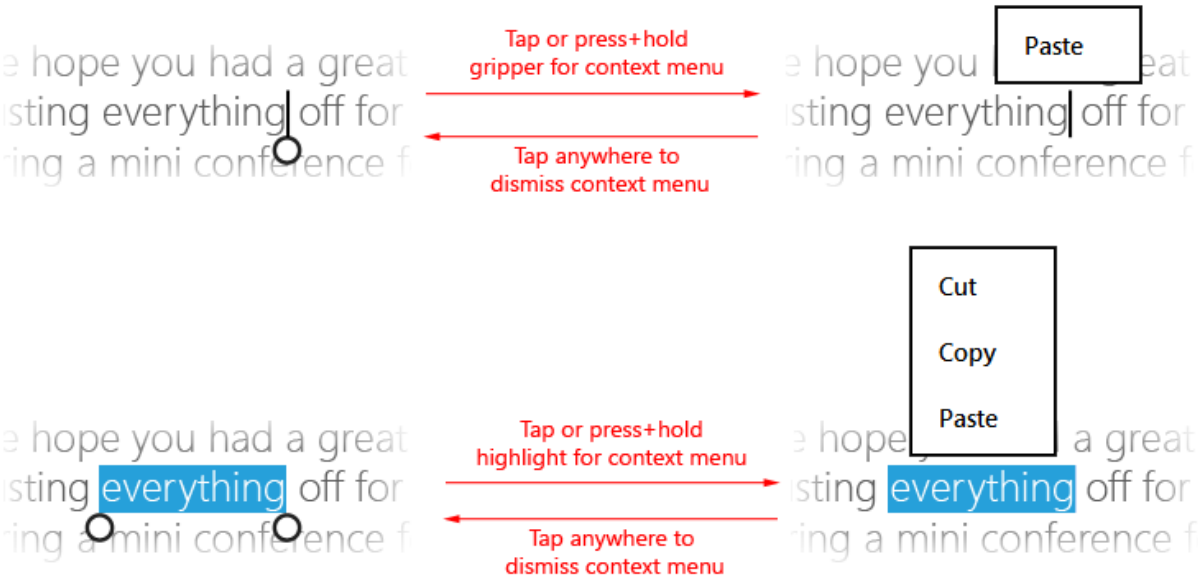
リッチ ツールチップのビジュアル デザインと情報は、標準のツールチップと区別しやすく、充実した内容にすることをお勧めします。

コンテキスト メニュー

コンテキスト メニュー (PopupMenu) は、ユーザーが Windows ストア アプリ内のテキストまたは UI オブジェクトに対する操作 (クリップボード コマンドなど) をすばやく実行できるようにするための簡易メニューです。

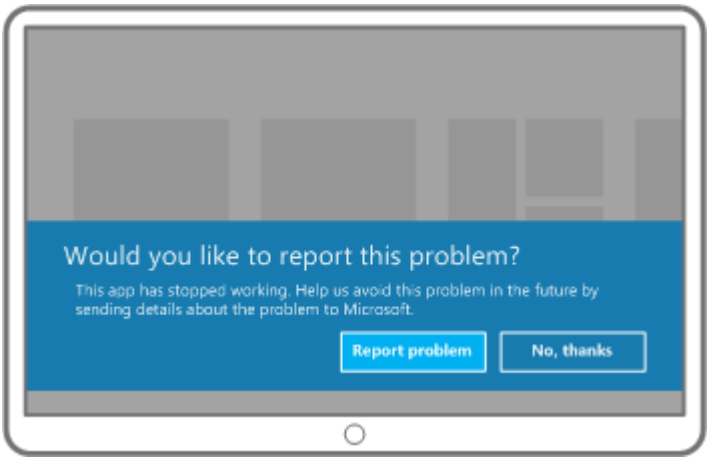
タッチ操作のメリットを活かしたコンテキスト メニューは、2 つの部分で構成されます。長押しによってビジュアルな合図 (ツールチップ) が表示されます。その後、ツールチップが消えてユーザーが指を離すと、コンテキスト メニュー自体が表示されます。

次の図は、選択範囲内またはグリッパー上でタップしてテキストの既定のコンテキスト メニューを呼び出す方法を示しています (長押しを使うこともできます)。



メッセージ ダイアログ

処理を続行する前に、ユーザーの操作やアプリの状態に基づいてユーザーに応答を求めるには、メッセージ ダイアログ (MessageDialog) を使います。明示的なユーザー操作が必須であり、ユーザーが応答するまで、アプリへの入力はブロックされます。



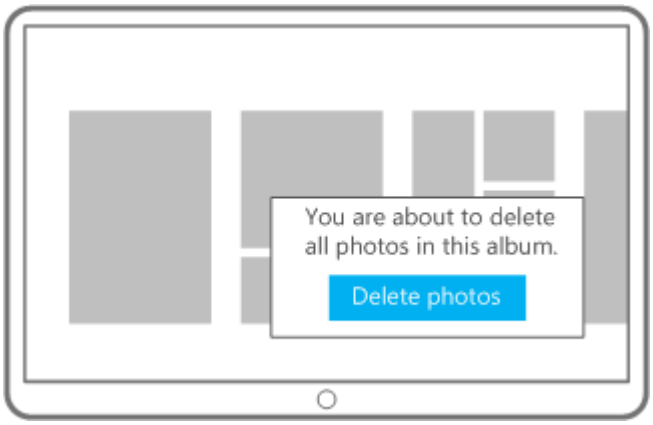
次に、メッセージ ダイアログを表示する一般的な理由をいくつか示します。

- 緊急の情報を提示する
- 実行を継続する前に質問する
- エラー メッセージを表示する

フライアウト

フライアウト (Flyout) は、タップ、クリック、またはその他のアクティブ化によって表示される軽量の UI パネルです。現在のアクティビティに関連する情報、質問、またはオプションのメニューをユーザーに表示するために使われます。簡易非表示にすることができます (ユーザーがフライアウト パネルの外部をタッチまたはクリックするか、Esc キーを押すと消えます)。つまり、フライアウトを無視できます。

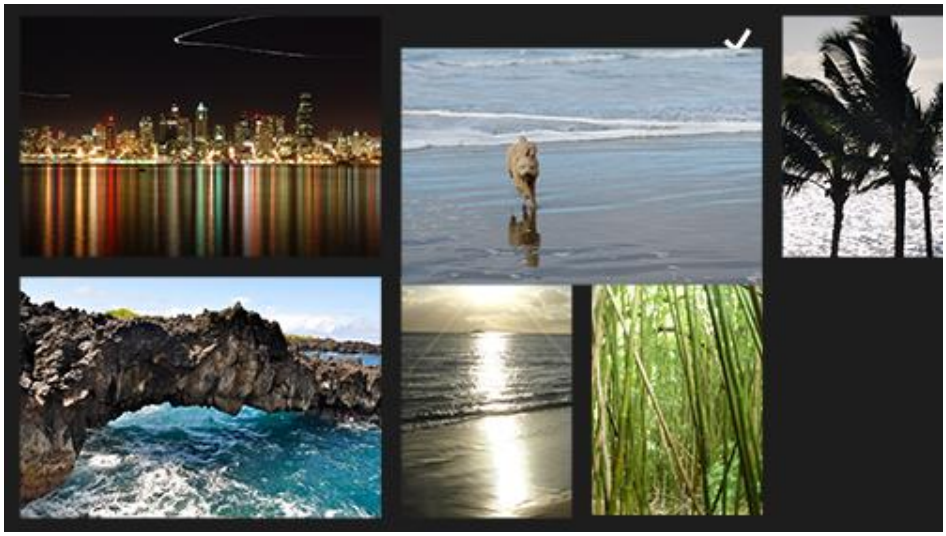
ツールチップとは異なり、フライアウトは入力を受け取ることができます。メッセージ ダイアログとは異なり、アプリはアクティブなままで、入力を受け取ります。



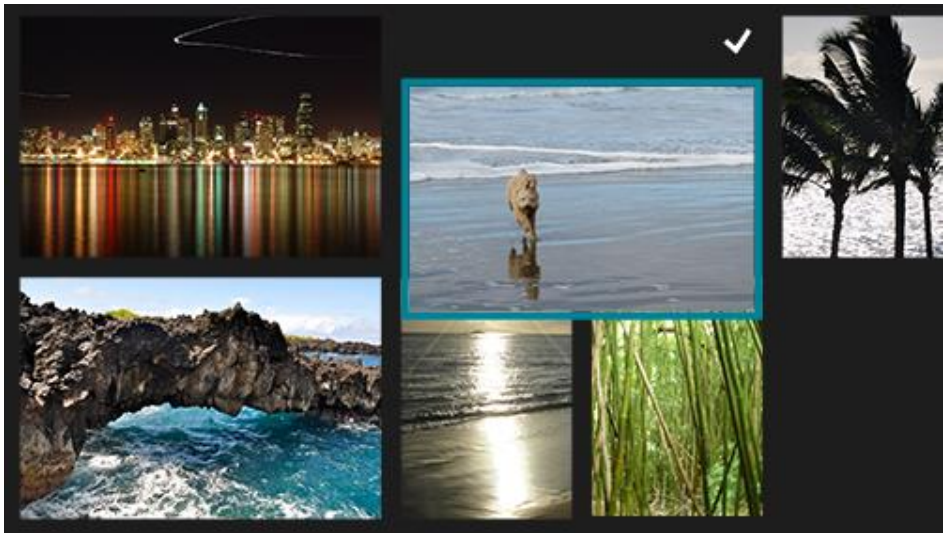
自己表明 UI

自己表明操作とは、ターゲット オブジェクトの操作方法を説明し、操作結果のプレビューを提供する、情報量の多いビジュアルなツールチップまたはアニメーションです。

次のいくつかの画像は、スタート画面でのクロススライド選択の自己表明操作を示しています。ユーザーがアプリのタイルに (ドラッグしないで) タッチすると、タイルが (ドラッグされたかのように) 下方向へスライドし、アプリを実際に選んだ場合に表示される選択チェック マークが現れます。



指で項目を押すと、選択内容に対する自己表明が始まります。自己表明により、項目に対して実行される操作が示されます。



指を離さずに、スワイプして実際に項目を選択します。

ユーザーがそのまま指をスライドすると、自己説明のビジュアル エフェクトが変わり、オブジェクトを移動できるようになったことが示されます。

自己表明が表示された後にユーザーが指を離すと、オブジェクトはアクティブでなくなります。

ヘルプと説明のガイドライン

ユーザーにヘルプやトラブルシューティングのヒントを提供して、効果的にアプリの操作法を示す必要があります。このセクションでは、アプリを使うユーザー向けの操作説明に関するベスト プラクティスを提供します。

このセクションの内容

トピック	説明
アプリのヘルプ	このガイドラインは、アプリの効果的なヘルプ コンテンツの設計方法を説明しています。
インストラクショナル UI	Windows ストア アプリを使う方法をユーザーに気付かせるユーザー インターフェイス (UI) をデザインします。

アプリのヘルプのガイドライン

このガイドラインは、アプリの効果的なヘルプ コンテンツの設計方法を説明しています。ヘルプ コンテンツは 1 ページに収めます。テキスト、リンク、画像を含めることができます。ダイナミックなヘルプ コンテンツを提供する場合は、サポート Web サイトにリンクするか、ヘルプ セクションにオンライン ページを埋め込みます。

アプリにヘルプ コンテンツを含めるかどうか

アプリにヘルプ セクションを含めるのは、必要な場合だけにします。たとえば、ユーザーが混乱する可能性のある UI 要素がアプリに 1 つか 2 つしかない場合は、インストラクショナル UI を組み込んだり、アプリ内の単純なデモを作成したり、UI 要素を設計し直したりして、わずかな問題のためだけにヘルプ ページを作成しないようにします。ただし、アプリがもっと複雑な場合は、有益なヘルプ コンテンツを追加するとユーザーにとっておおいに役立つ可能性があります。

推奨と非推奨

- ヘルプ コンテンツは設定ポップアップを使って表示します。設定フライアウトを使うと、Windows 標準の設定ウィンドウに簡単にヘルプを追加できます。

- 設定ウィンドウからヘルプにアクセスできるようにします。
- 設定ウィンドウ内のエントリ ポイントのラベルを [ヘルプ] にして、明確にわかるようにします。推奨事項について詳しくは、「アプリ設定のガイドライン」をご覧ください。
- ヘルプ ページは短くしてユーザーが目を通しやすくします。
- ヘルプ コンテンツが 1 ページに収まらない場合、または更新が必要な情報をダイナミック コンテンツに含める必要がある場合は、サポート Web サイトにリンクするか、ヘルプ フライアウトにオンライン ページを埋め込みます。Web ページにリンクすると、ユーザーがアプリから引き離されることを忘れないでください。可能な場合は、オンライン コンテンツを埋め込んで、より統一感のあるユーザー エクスペリエンスを実現します。JavaScript を使ったアプリを作成している場合は、「オンライン コンテンツをヘルプに含める方法」で詳しい手順をご覧ください。
- 設定ウィンドウのヘルプのエントリ ポイントから Web サイトに直接リンクしないでください。代わりに、関連付けられたヘルプ ポップアップでオンライン コンテンツへのリンクを示します。
- 専門的な用語や業界用語を使わないでください。
- ヘルプを使って、アプリのあらゆる機能を説明しないでください。ヘルプは、基本的な事項をすばやく参照できる場所にします。アプリについて詳しく説明したい場合は、ヘルプ コンテンツの最後に、サポート Web ページへのリンクを記載することを検討してください。
- アプリの新しいバージョンが利用できることをユーザーに知らせるためにヘルプを使わないでください。バージョン情報は、既定で、設定ウィンドウの **[アクセス許可]** セクションに表示されます。

インストラクショナル UI のガイドライン

Windows ストア アプリを使う方法をユーザーに気付かせるユーザー インターフェイス (UI) をデザインします。

推奨と非推奨

- アプリでできることを新しいユーザーに紹介するためにインストラクショナル UI を使います。
- 新機能に関するヒントや更新後のアプリの変更点に関する詳細に使います。
- インストラクショナル UI を特定のタスクと統合します。
- アプリケーション UI の操作を妨げないようにします。

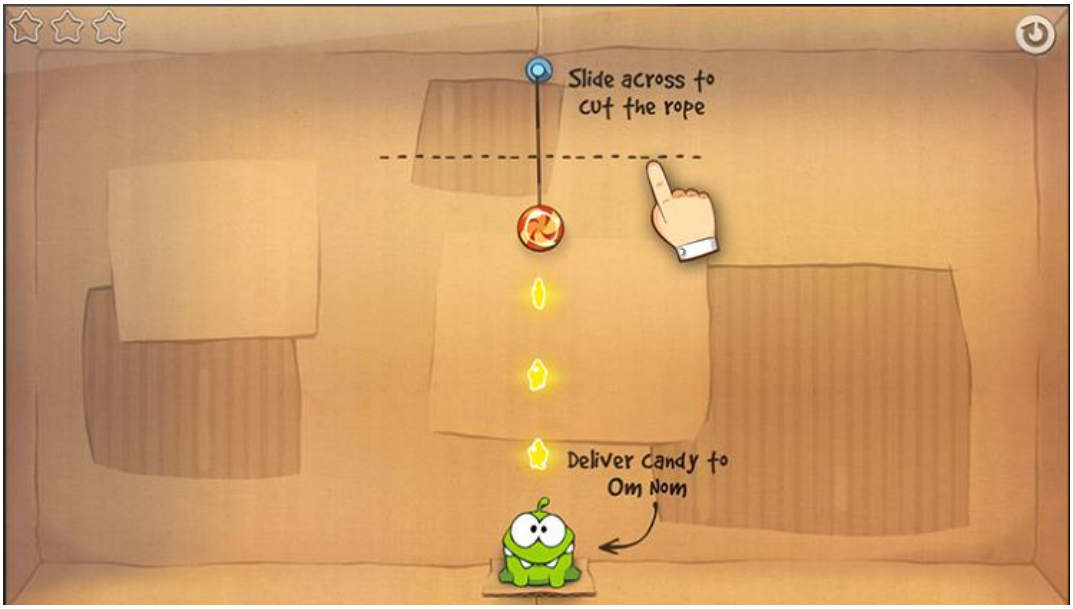
その他の使い方のガイダンス

状況によっては、ユーザーが Windows ストア アプリを操作できるようにする最適な方法は、アプリの UI によって気付かせることです。この種のガイダンスは、インストラクショナル UI という用語で表します。良い例の 1 つとして、タスクを完了するためにタッチ操作を使う必要があるときに、ユーザーへの指示のためにインライン テキストやフライアウトなどの UI 要素を使うことが挙げられます。

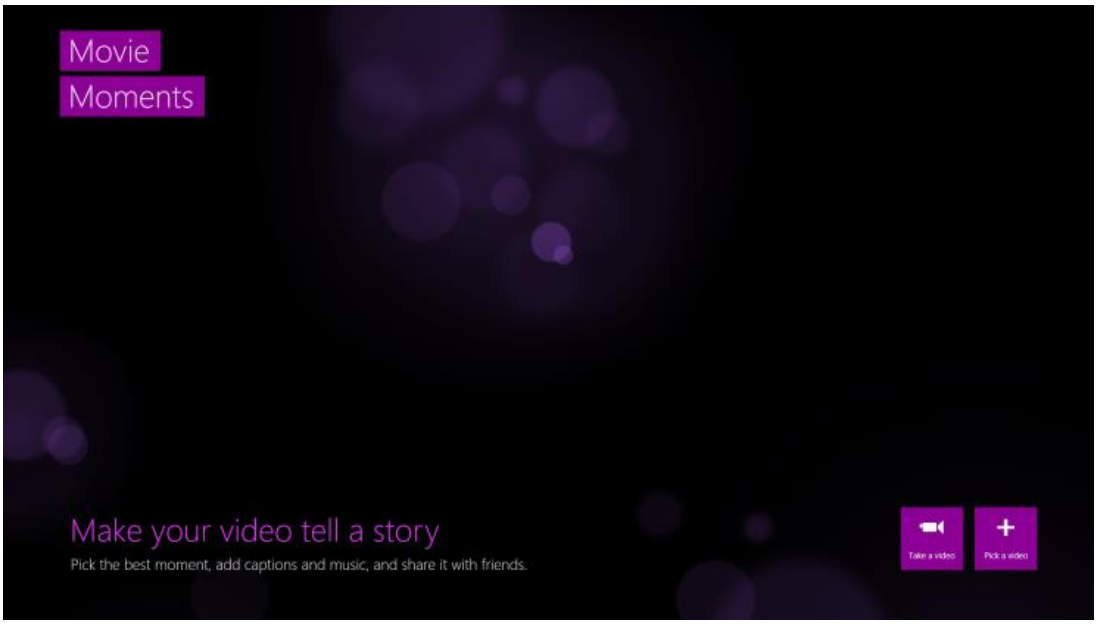
インストラクショナル UI はよく練られた設計に代わるものではないことに注意してください。使用頻度が多すぎる場合、またはコンテキストから逸脱する場合には、アプリのフローが中断され、実効性が低下するおそれがあります。インストラクショナル UI を追加する前に、ユーザーをアプリに導く他の方法を検討してください。

ここでは、インストラクショナル UI がユーザーが操作を習得するのに役立ついくつかの例を示します。

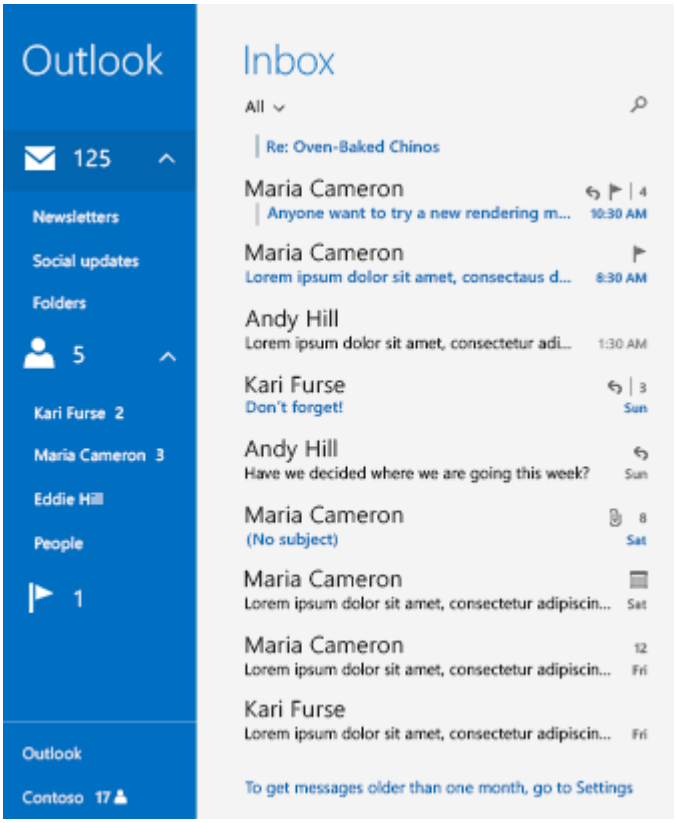
- **ユーザーがタッチ操作を見つけられるようにする。** 次のスクリーン ショットには、Cut the Rope というゲーム内でタッチ ジェスチャーを使う方法をプレイヤーに気付かせるインストラクショナル UI が示されています。



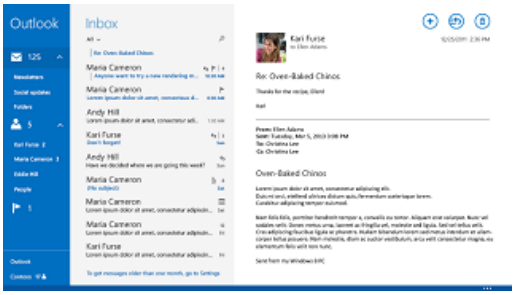
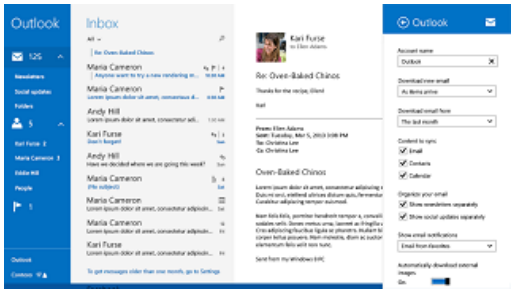
- **第一印象を良くする。** 新しいユーザーにアプリで可能な処理を紹介するためにインストラクショナル UI を使うことを考慮してください。たとえば、ムービー モーメントの初回起動時には、インストラクショナル UI を使ってユーザーは映画を作成するように求められます。



- **複雑なタスクの次の手順にユーザーを導く。**Windows メール アプリの受信トレイの下のヒントは、以前のメッセージにアクセスできるように、[設定] にユーザーを導きます。



ユーザーがメッセージをクリックすると、アプリの設定フライアウトが画面の右側に表示され、ユーザーがタスクを実行できるようになります。次のスクリーン ショットは、ユーザーがヒントのメッセージをクリックする前後のメール アプリを示しています。

クリック前	クリック後
	

- **UI の変更を示す。**最新バージョンのアプリの UI に大幅な変更が加えられた場合、ユーザーはアプリの新しい機能や変更の詳細についてのヒントを希望することが考えられます。

インストラクショナル UI の設計原則

- **シンプルな状態を保つ。**一度に 1 つの基本的な概念を紹介し、可能な限り図を使います。複雑な機能について説明する場合は、Windows ストア アプリの設定フライアウトにヘルプ セクションを追加することを検討してください。
- **コンテキストの中で説明する。**インストラクショナル UI をタスクに統合します。そうすれば、ユーザーがタスクを完了しやすくなります。最も必要としているときに紹介された概念は、ユーザーの記憶に残りやすいものです。
- **操作を妨げない。**インストラクショナル UI が表示されている間も、アプリを操作し続けられるようにします。インストラクショナル UI は、ユーザーの邪魔になることなく、有用であることが必要です。
- **説明が終わったら消える。**説明が終わったらすぐにインストラクショナル UI が消えるようにするか、ユーザーが終了できるようにします。 また、ほとんどの場合、インストラクショナル UI は一度表示されれば十分です。繰り返し同じインストラクショナル UI を表示することは避けてください。
- **控え目に使う。**設計とヘルプ セクションが練られていれば、新しいアプリを利用するために把握する必要があることを、ユーザーに十分に伝えることができます。アプリにインストラクショナル UI を追加する前に、さまざまな設計オプションを検討してください。

レイアウトとスケーリングのガイドライン

このセクションでは、アプリの各ページでのアプリ要素のレイアウトと、さまざまなサイズのデバイスやサイズ変更されたウィンドウでユーザーがアプリを操作する際の、アプリのスケーリングに関するガイドラインを示します。また、アプリのレイアウトに広告を統合するためのガイドラインも示します。

このセクションの内容

トピック	説明
広告	アプリ開発者にとって、広告は収益を獲得するための重要な手段です。また、Windows 8 の幅広いリーチは、広告主にとっても魅力的な機会です。
複数のウィンドウ	Windows ストア アプリで複数のウィンドウをサポートする場合は、次の推奨事項に従ってください。
プロジェクション マネージャー	プロジェクション マネージャーを使うと、アプリの個別のウィンドウを別の画面にプロジェクションできます。
幅の狭い レイアウト	ユーザーが縦長、つまり狭いビューにサイズを変更した場合にアプリの UI が調整されるように設計します。
ピクセル密度に 合わせた スケーリング	さまざまなピクセル密度のデバイスに合わせて調整する場合は、アプリの UI の品質を維持するために、次のガイドラインに従ってください。
複数の画面サイズ のサポート	すべてのサイズと向きの画面に対応するように UI を設計する際は、次のガイドラインに従ってください。

広告のガイドライン

アプリ開発者にとって、広告は収益を獲得するための重要な手段です。また、Windows 8 の幅広いリーチは、広告主にとって魅力的な機会です。広告一般について詳しくは、Ads 101 のドキュメントをダウンロードしてください。

例

ここでは、特定のアプリ レイアウトと一体化するようにデザインされた広告の例を 2 つ示します。最初の画像は、全画面グリッド レイアウトのアプリを示しています。ご覧のように、広告はコンテンツと同じサイズ、同じ形状です。

次の画像では、アプリは幅の狭いウィンドウに表示されています。広告は、コンテンツと同様に、幅の狭いレイアウトに合わせて調整されています。

推奨と非推奨

アプリに広告を組み込むときは、次の推奨事項に従ってください。

- デザインに広告を組み込みます。元のデザインやコンテンツ レイアウトなど、アプリ全体に広告を組み込んで、統一されたエクスペリエンスを生み出します。アプリの主要な部分から細かな部分まで、どれくらいのスペースを広告に割り当てるかを検討してください。
- 補完的な広告フォーマットを選びます。アプリに適した広告フォーマットを選びます。利用可能な広告フォーマットは多彩ですが、ユーザーを引き込み、シームレスなエクスペリエンスを提供するフォーマットを選ぶ必要があります。
- 補完的な広告サイズと配置場所を選びます。アプリのデザインを補完し、業界標準に従っていて、あらゆる広告主にとって使い勝手の良い広告サイズと配置場所を選びます。 たとえば、グリッド レイアウトの場合、250 x 250 ピクセルの広告を使って、グリッド内に広告を配置することを検討します。広告が収まらない場合は、別のクラスターの作成を検討します。 広告サイズのその他の例については、Microsoft Advertising SDK for Windows 8 のページをご覧ください。
- すべてのウィンドウ サイズ向けにデザインします。アプリのウィンドウ サイズと向きに応じて、広告の表示領域がどのように変化するかを検討します。

- ローカル広告を検討します。特定の地域向けの広告 (ローカル広告) を提供するかどうかを検討します。
- キーワードを組み込みます。特定のターゲット向けに発信する広告主のために、アプリの分類に役立つキーワードを組み込みます。
- 実績のある指標を使います。広告を有効にする際に、収益を獲得できるかどうか、CPM (1000 回の広告表示あたりの料金) とフィル レート (広告の表示率) の 2 つの主な指標から判断します。CPM にフィル レートを乗算した値は、収益を判断するうえで最適なソリューションと考えられています。
- アプリで広告だけを表示しないでください。アプリに広告を含める場合、単なる広告にとどまらない機能を提供する必要があります。また、ユーザーがアプリ内で主要なタスクを完了できるようにする必要があります。アプリは単に Web サイトを開くもの、または単に Web サイトの動作を模倣するものにはしないでください。
- Windows ストアのコンテンツ ポリシーに違反する広告は掲載しないでください。コンテンツ ポリシーは、「Windows ストア向けアプリ認定要件」の第 5 項で説明されています。
- アプリの説明、タイル、通知、アプリ バー、または端からのスワイプ操作を使って、広告を表示しないでください。

広告の具体的な要件は、「Windows ストア向けアプリ認定要件」の第 2 項で説明されています。これらの要件に従えば、ユーザー エクスペリエンスを損なうことなく、アプリに広告を組み込むことができます。

その他の使い方のガイダンス

アプリの広告主を選ぶときは、次の基本的な質問に答えてください。

- 広告主は Windows ストア向けアプリ認定要件に従っていますか?
- 広告主はご自身のアプリにふさわしい内容の広告を提供していますか?

複数のウィンドウのガイドライン

複数のウィンドウのサポートによって、ユーザーはアプリの異なる部分を同時に操作できます。ユーザーは、複数のウィンドウを使って、コンテンツを比較できるほか、コンテンツの複数の特定部分を同時に表示できます。Windows ストア アプリで複数のウィンドウをサポートする場合は、次の推奨事項に従ってください。

説明

複数のウィンドウをサポートするアプリでは、各ウィンドウが別個のアプリのように動作します。ユーザーがスタート画面でアプリのタイルをクリックすると、一番最後に使ったアプリのウィンドウが表示されます。ユーザーはウィンドウごとにサイズを変更したり、ウィンドウを個別に非表示にしたりできるほか、最近使ったアプリの一覧に個々のウィンドウを表示することもできます。

複数ウィンドウの設計

複数のウィンドウをサポートすることがアプリにとって適切な場合、各ウィンドウに表示するコンテンツを決定する必要があります。たとえば、1 個のメイン ウィンドウと、限定された固有の機能セットを持つセカンダリ ウィンドウを作ることも、元のアプリ ウィンドウのコピーとして新しい個別のウィンドウを設計することもできます。ユーザーがアプリを切り替えたときに表示される、セカンダリ ウィンドウのタイトルを指定することもできます。

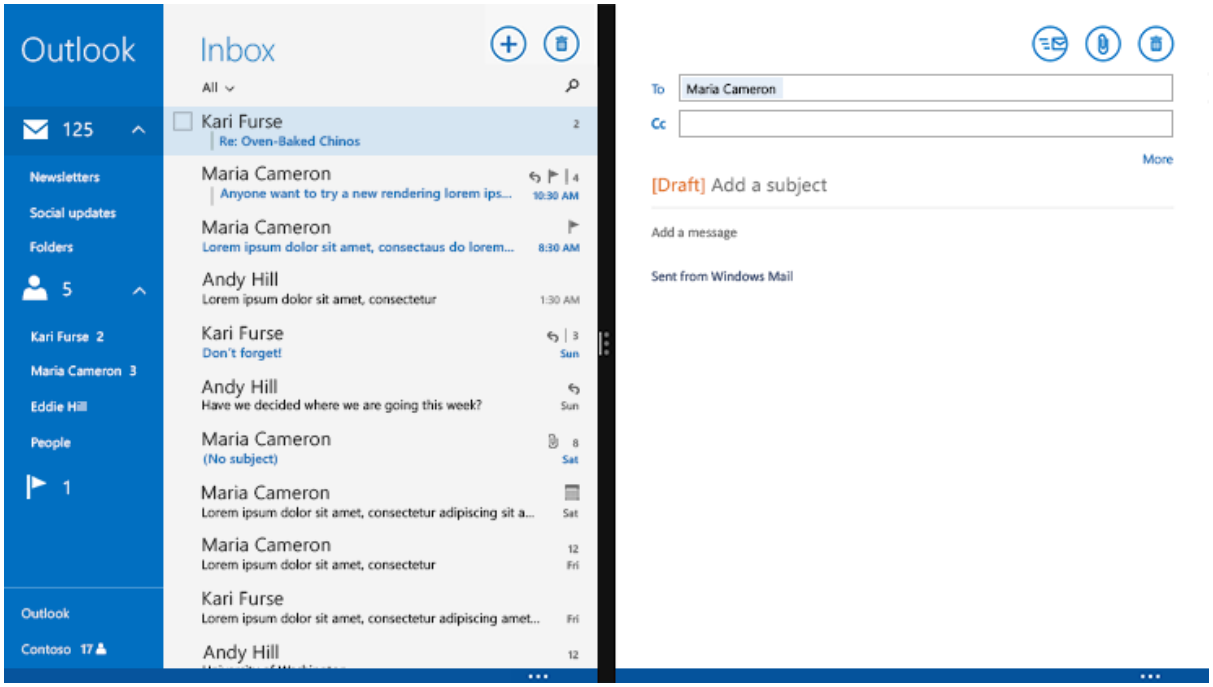
また、(元のアプリ ウィンドウを基準にして) 新しいウィンドウを開く画面上の位置も指定します。新しいウィンドウは、次のいずれかの位置に配置できます。

- 元のウィンドウの隣で、画面の領域を共有する。
- メイン ウィンドウの位置に表示する。
- 画面にまったく表示されない。

セカンダリ ウィンドウが最初に表示された後は、ウィンドウの配置とサイズをユーザーが制御できます。

例

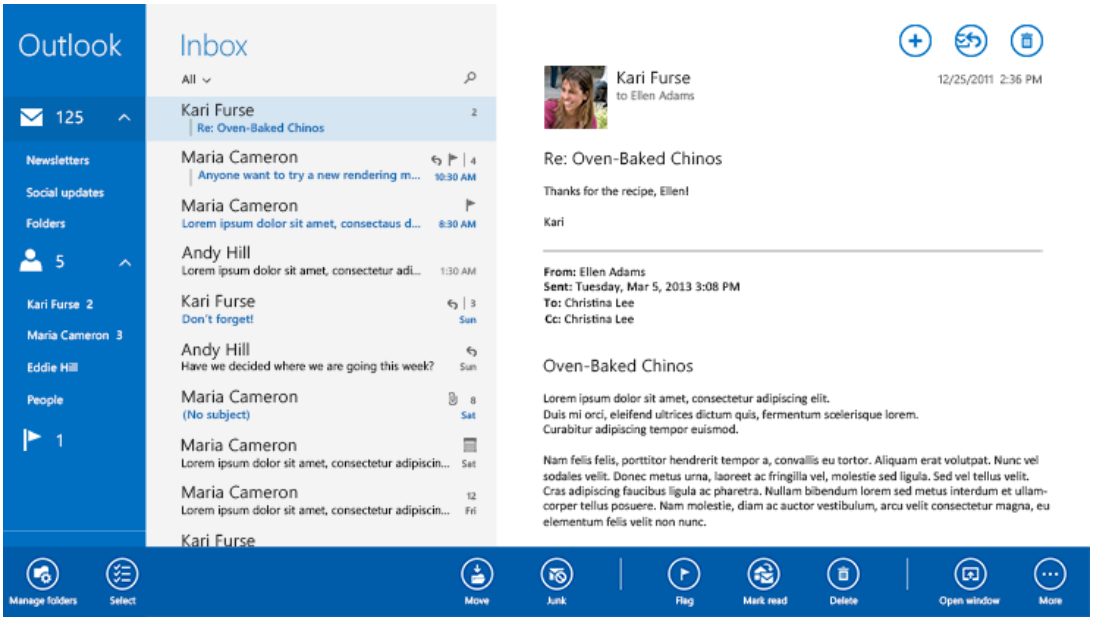
複数のウィンドウを使うアプリの例としてメール アプリがあります。ユーザーは、メイン アプリ ウィンドウでメッセージを表示したり、新しいウィンドウを開いたりすることができます。このことは、たとえば、新しいメッセージを作成しながら、同時にメイン ウィンドウを使って他のメッセージを検索する場合に便利です。



メール アプリで 2 つのウィンドウが開いている場合、最近使ったアプリの一覧が次のように表示されます。

推奨と非推奨

- セカンダリ ウィンドウからメイン ウィンドウに戻る方法をユーザーに提供します。
- ユーザーが新しいウィンドウを開くための明確な操作を提供します。たとえば、新しいウィンドウを開くボタンをアプリ バーに追加します。メール アプリの一番下のアプリ バーに **[開く]** ボタンが表示されます。



- 新しいウィンドウには、必ずそのウィンドウのコンテンツを反映したタイトルを付けます。ユーザーがタイトルに基づいてアプリの各ウィンドウを区別できるようにします。
- **consolidated event** に受信登録し、イベントが発生したときにはウィンドウの内容を閉じます。統合イベントは、最近使ったアプリの一覧からウィンドウが削除されたときや、ユーザーがウィンドウを閉じるジェスチャを実行した場合に発生します。
- 元のアプリ ウィンドウを新しいウィンドウで置き換える場合は、ウィンドウの切り替え時にカスタム アニメーションを提供します。
- 生産性を向上させ、マルチタスクを可能にするシナリオでは、アプリ内で新しいウィンドウを有効にします。
- 新しいウィンドウは、ユーザーがウィンドウ内でタスク全体を完了できるように設計します。
- ユーザーがアプリの別の部分に移動したときに新しいウィンドウを自動的に開かないでください。新しいウィンドウは、常にユーザーの操作によって開くようにします。
- アプリの主要な目的を完了するために、新しいウィンドウを開くことをユーザーに要求しないでください。

プロジェクション マネージャーのガイドライン

プロジェクション マネージャーを使うと、アプリの個別のウィンドウを別の画面にプロジェクションできます。 たとえば、ゲーム アプリでは、大型のモニターにメインのゲーム プレイ画面を表示し、ローカルの画面にゲームのコントロールを表示できます。また、Scrabble などのマルチプレイヤーのワード ゲームでは、プロジェクション画面に共有ゲーム ボードを表示し、ローカルの画面にユーザーのゲーム ピースを表示できます。プレゼンテーション アプリの場合、プロジェクション画面のウィンドウにプレゼンテーションを表示し、ローカルの画面に発表者のノートを表示できます。

既定では、プロジェクション マネージャーを使わない状態で、ユーザーが他の表示デバイスに接続している場合、アプリのウィンドウがサブ画面に複製されます。複製モードのとき、両方の画面に適した解像度が Windows によって自動的に選ばれます。外部画面の画面情報は使われません。しかし、この解像度が動画再生やゲームに最適でないことがあります。プロジェクション マネージャーを使うと、Windows によってプロジェクション画面の解像度と画面の縦横比が取得され、ウィンドウの表示が最適化されます。

プロジェクション マネージャーは、アプリ用に複数のウィンドウを使うことに似ています。この表では、複数のウィンドウとプロジェクション マネージャーの使い分け方について説明します。

シナリオ	複数のウィンドウを使う	プロジェクション マネージャーを使う
ユーザーが両方のウィンドウを操作	推奨	非推奨 (ただし外部ディスプレイが Perceptive Pixel (PPI) by Microsoft のようなタッチ デバイスである場合は除く)
サブ ウィンドウが操作用ではなく表示専用	非推奨	推奨
ユーザーが縦横比や解像度が大きく異なる画面にサブ ウィンドウを表示	非推奨	推奨

推奨と非推奨

- ユーザーがローカルのアプリ ウィンドウからプロジェクションを制御する。

  ユーザーが次の操作を実行できる必要があります。

  ◦ 新しいプロジェクション ウィンドウを起動する。

  ◦ ウィンドウの中断したプロジェクションを再開する。

  ◦ 開始したプロジェクションを停止する。

  ◦ プロジェクション ウィンドウ上のコントロールを使ってローカルとプロジェクションのウィンドウを入れ替える。ウィンドウの自動配置が適切でなく、プロジェクション ウィンドウがローカルの画面に表示される場合、ユーザーはウィンドウを入れ替えることができる必要があります。

- 次のアイコンを使って、プロジェクションの開始、停止、ウィンドウの入れ替えを行う。

コード	アイコン	説明
U+E2B4		プロジェクションの開始または再開
U+E2B3		プロジェクションの停止
U+E13C		プロジェクション ビューの入れ替え

- プロジェクションを自動的に開始または停止しない。プロジェクションはユーザー入力によってのみ開始または停止する必要があります。

  注 "再開" 機能を実装することで、一時停止後や他のアプリへの切り替え後にユーザーが簡単にプロジェクションを再開できるようにすることが可能です。プロジェクション ウィンドウが表示中の画面から切り替わった場合 (通常、別のアプリのプロジェクションのため)、**StartProjectingAsync** を使ってプロジェクション ウィンドウの表示を再開します。**VisibilityChanged** イベントにサブスクライブして、いつプロジェクション ウィンドウが表示中の画面から切り替わったかを調べることができます。**consolidated** イベントにサブスクライブして、いつプロジェクション ウィンドウが最近使ったアプリの一覧から削除されたか、いつプロジェクション ウィンドウが閉じられたかを調べることができます。

その他の使い方のガイダンス

スタイルとレイアウト

プロジェクションの開始、停止、入れ替えに使用するアイコンの色やラベル テキストは選ぶことができます。アイコンの配置場所を選ぶことができますが、アイコンは下部のアプリバーに配置し、アプリ バー ボタンのガイダンスに従うことをお勧めします。

プロジェクション ウィンドウの配置を変更することはできません。自動的に決定されるためです。マウス、キーボード、またはタッチパッドを操作するユーザーはプロジェクション画面を配置後に移動できます。

幅の狭いレイアウトのガイドライン

ユーザーが縦長、つまり狭いビューにサイズを変更した場合にアプリの UI が調整されるように設計します。そのように計画した場合はこのトピックのガイドラインが適用されます。

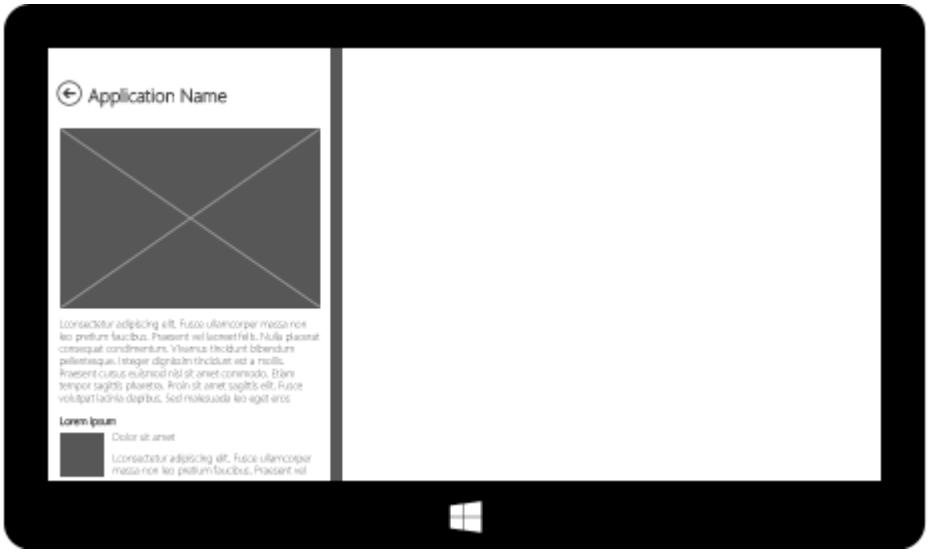
- アプリの最小幅を既定の 500 ピクセルでなく 320 ピクセルに変更します (幅の狭いレイアウト)。
- ユーザーがアプリのサイズを、高さが幅よりも大きくなるように (縦長のレイアウト) 変更したときに、縦方向のレイアウトに切り替わるようにアプリを設計します。

一般的な推奨事項について詳しくは、「複数の画面サイズをサポートするためのガイドライン」をご覧ください。狭いレイアウト向けのゲームの設計に関する具体的なガイダンスについては、ゲームのアイデア ブックをご覧ください。

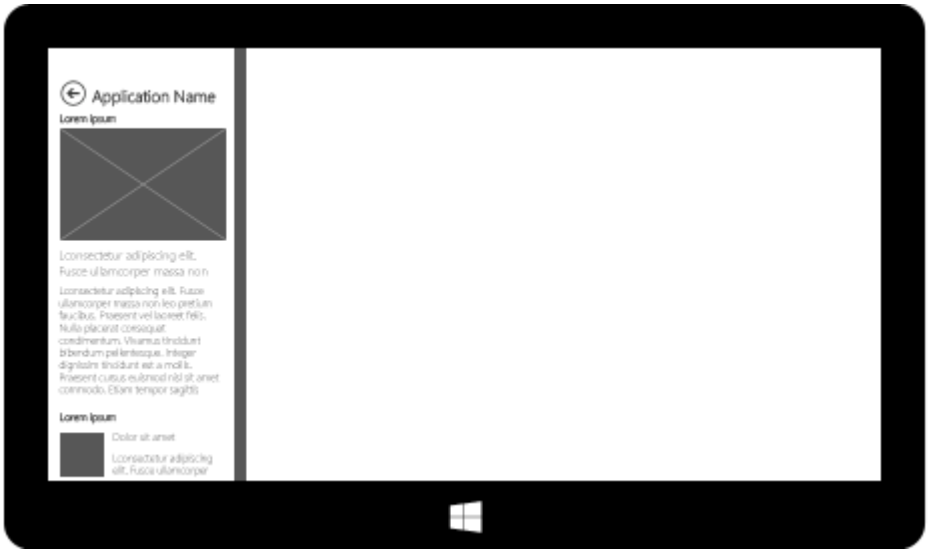
例

幅の狭いレイアウト

既定では、Windows ストア アプリの最小幅は 500 ピクセルです。ここに幅が 500 ピクセルのアプリがあります。



もう 1 つ、幅が 320 ピクセルのアプリがあります。

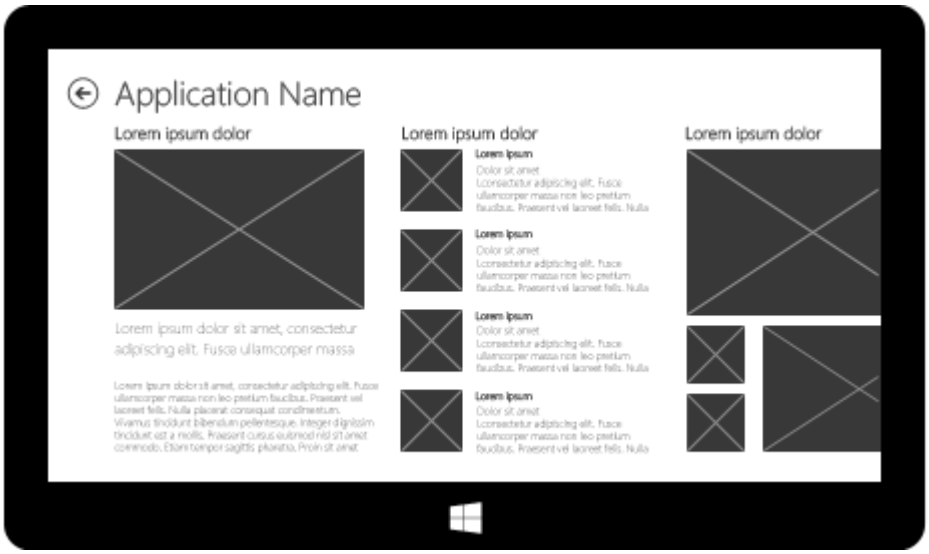


500 ピクセル未満の最小幅をサポートする場合は、狭い幅でもアプリの機能を使うことができるように設計を少し変更する必要があります。500 ピクセル未満の幅でアプリを有効に動作させるには、「推奨と非推奨」に従ってください。

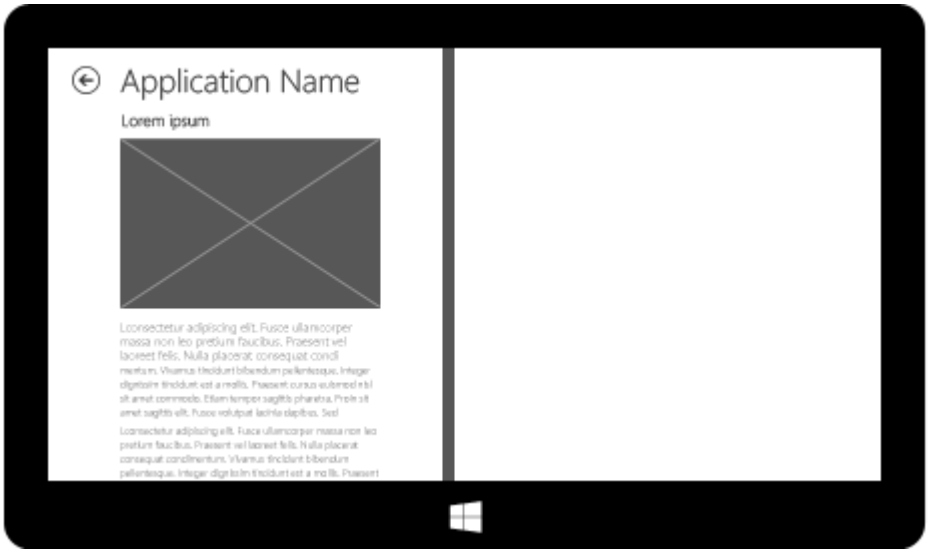
縦長のレイアウト

さらに、アプリの幅より高さが大きくなったときにアプリのデザインを変更することもできます。たとえば、アプリの幅より高さが大きくなったときに、水平方向ではなく垂直方向にパンするようにアプリを設計できます。

全画面表示のときに水平方向にパンするアプリを次に示します。



これが、幅よりも高さが大きい状態のアプリです。この場合、アプリは垂直方向のパンします。



320 ピクセルの最小幅をサポートするか

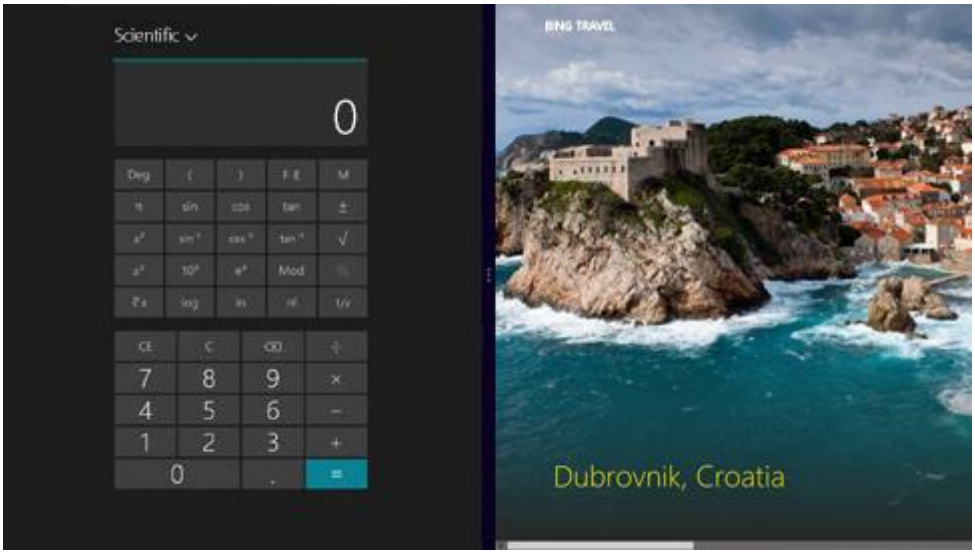
既定の最小幅よりさらに狭い幅をサポートするかどうかは、ユーザーがアプリでどのような作業を行うかによって判断が分かれます。狭い幅として 320 ピクセルまでサポートする必要がある一般的なシナリオには、次のようなものがあります。

- アプリにとってマルチタスクが重要である。
- ユーザーがアプリを画面上に維持することが望ましい。
- アプリがコンパニオン シナリオで他のアプリと共に動作する。
- アプリが狭い幅でうまく動作する。

推奨と非推奨

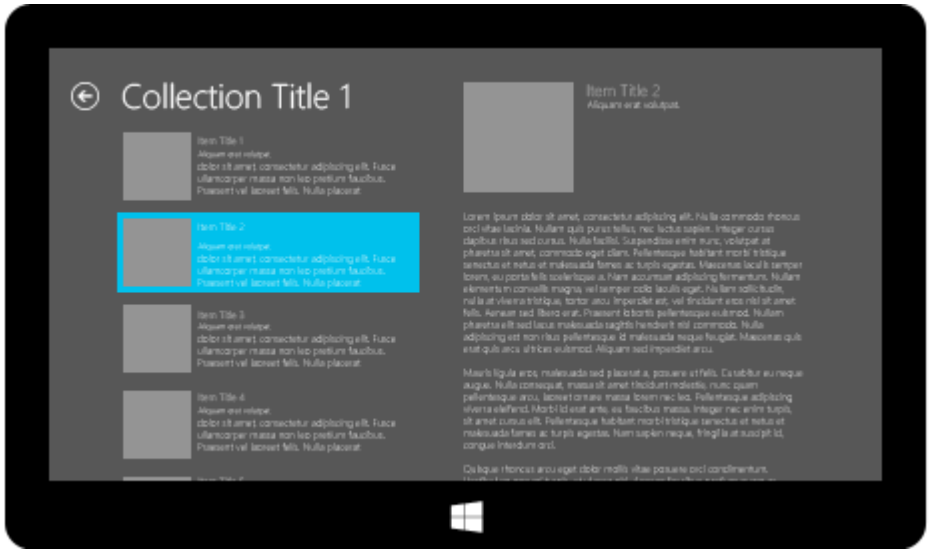
- アプリが全画面表示のときに水平方向にパンしている場合、アプリ ウィンドウが縦長のときには垂直方向のパンに切り替えます。
- アプリの幅が 500 ピクセル未満の場合、狭いサイズに対応するために次のデザイン変更を行います。
  - 小さい戻るボタン スタイルを使います。戻るボタンのサイズについて詳しくは、「Segoe UI Symbol アイコンのガイドライン」をご覧ください。
  - 左余白は 20 ピクセルにします。
  - アプリのヘッダーには 20 ポイントのサイズを使います。
  - ページ切り替えアニメーションとコンテンツ切り替えアニメーションには、より小さなオフセット値を使います。

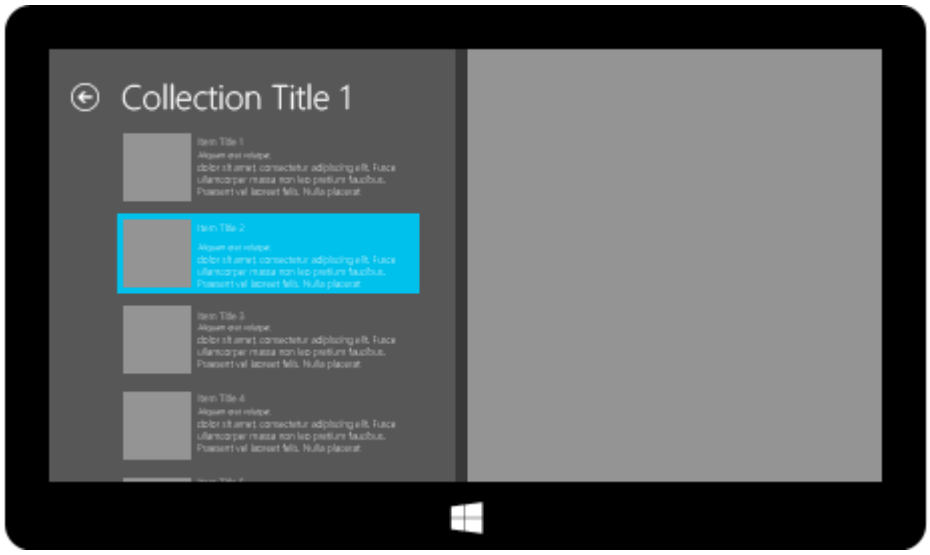
これは通常幅の電卓アプリです。



そしてこちらは、同じアプリですが、幅が 320 ピクセルです。左余白が 20 ピクセルになり、ヘッダーのフォントは幅の狭いレイアウトの推奨事項に従って、20 ポイント サイズに縮小しました。

- アプリが全画面表示のときに垂直方向にパンしている場合、アプリの高さが幅より大きい場合はサイズが 1 つの列または 1 ペインに縮小します。アプリが 1 つの列またはペインに切り替えられる正確な幅を決定します。1 つの列または 1 つのペインのビューには、必ずユーザーがペイン間を移動するためのナビゲーションを含めます。





- アプリのレイアウトとすべてのコントロールが、最小サイズまで縮小され、縦長で幅が狭いアプリ ウィンドウで使用できるように設計します。考慮する必要がある重要なコントロールは、次のとおりです。
  - 上部と下部のアプリ バー
  - メッセージ ダイアログ
  - フライアウト
  - 設定ウィンドウ
- ウィンドウのサイズが狭い幅に変更されたときに、ユーザーをアプリの別の部分に移動しないでください。
- 幅の狭いサイズでアプリの機能のほとんどが保持できない場合は、既定の最小値 (500 ピクセル) より狭い幅をサポートしないようにします。

ピクセル密度に合わせたスケーリングのガイドライン

Windows ランタイム アプリ (Windows、Windows Phone、またはその両方で実行) はシステムによって自動的に調整されて、画面のピクセル密度に関係なく一貫した読みやすさと機能を実現します。さまざまなピクセル密度のデバイスに合わせて調整する場合は、アプリの UI の品質を維持するために、次のガイドラインに従ってください。

注 次のガイドラインは、Silverlight を使った Windows Phone アプリには適用されません。Silverlight の具体的なガイダンスについては、Multi-resolution apps for Windows Phone 8 をご覧ください。

Standard slate
10.6" 1366x768
148 DPI

HD slate
10.6" 1920x1080
208 DPI

説明

スケーリングを使わない場合、ディスプレイ デバイスのピクセル密度が増すと、画面上のオブジェクトの物理的なサイズが小さくなります。UI がタッチできないほど小さくなったり、テキストが読み取れないほど小さくなったりする場合、Windows はシステムとアプリの UI を次のスケール プラトーの 1 つにスケーリングします。

Windows ストア アプリ :

- 1.0 (100%、スケーリングの適用なし)
- 1.4 (140% のスケーリング)
- 1.8 (180% のスケーリング)

Windows Phone ストア アプリ：

- 1.0 (100%、スケーリングなし)
- 1.4 (140% のスケーリング)
- 1.8 (180% のスケーリング)

Windows は、物理画面サイズ、画面解像度、画面の DPI、フォーム ファクターに基づいて、どの表示スケール プラトーを使うかを決定します。画面の仕様が特定のしきい値を満たす場合、次に高い表示スケール プラトーが使用されます。スケール ファクターを判断するためには、**ResolutionScale** または **RawPixelsPerViewPixel** (Windows Phone のみ) を使用できます。

推奨と非推奨

- スケーラブルなベクター グラフィックスを使います。 Windows はこれらの形式を自動的にスケーリングします。特に目に付くようなアーティファクトは生まれません。JavaScript アプリでは SVG を使います。C#、C++、または Visual Basic を使うアプリで XAML で定義されたグラフィックスを使用することができます。
- アプリ パッケージのビットマップ イメージにはリソース読み込みを使用し、各倍率に異なるイメージを用意します。画像ファイル名 (たとえば、Assets¥Square7070Logo.scale-100.png) に倍率を含めます。Windows は、自動的に現在のスケールに対して適切な画像を読み込むことに注意してください。DPI に応じたスケーリングのサンプルは、画像のリソース読み込みの使用法を示しています。

  アプリの認定に関連する画像要件については、「アプリの画像の選択」をご覧ください。命名規則について詳しくは、「クイック スタート: ファイルまたは画像リソースの使用 (JavaScript と HTML を使った Windows ストア アプリ)」または「クイック スタート: ファイルまたは画像リソースの使用 (C#/VB/C++ と XAML を使った Windows ストア アプリ)」をご覧ください。
- さまざまなスケール プラトーの資産を作成した場合:
  - ビットマップ イメージを 100% で設計して、手動で拡大しないでください。高品質の画像プログラムを使っていても、画像はぼやける可能性があります。

    - 大きな、高解像度の画像を縮小しても、必ずしも鮮明でくっきりしたものにはならないことにご注意ください。ただし、元のベクターが利用できない場合は、低解像度のファイルを拡大するよりは手動で高解像度のファイルを縮小する方が良い方法です。
- アプリが実行時にコードを使って画像を読み込む場合 (たとえば XAML や HTML ではなく DirectX を使って UI を作成している場合) は、**ResolutionScale** または **RawPixelsPerViewPixel** (Windows Phone のみ) を使ってスケールを特定し、スケールの割合に基づいて画像を手動で読み込みます。
- ファイル システム イメージに **Thumbnail** API を使います。サムネイル API では、サムネイルとして使用する小さな画像をキャッシュしてパフォーマンスを最適化します。詳しくは、「ファイル システムへのアクセスの効率化」をご覧ください。
- 大きな画像が読み込まれたときにレイアウトが変わらないように、自動サイズ設定を使わずに画像の幅と高さを指定します。
- タイポグラフィ グリッド単位とサブ単位を使います。メジャー グリッド単位にはタイポグラフィ グリッドで定義された 20px のサイズを使い、マイナー グリッド単位には 5px を使って、ピクセルの丸めによりレイアウトでピクセル シフトが生じないようにします。5px で割り切れるサイズ単位の場合、ピクセルの丸めは行われません。
- リモート Web 画像には解像度メディア クエリを使います。 アプリが JavaScript を使用し、リモート Web 画像がある場合は、CSS の @media 解像度メディア機能を **background-image** プロパティで使って、画像を実行時に置き換えます。
- 5px の倍数に調整されていない画像を使わないでください。5px の倍数以外の単位を指定すると、140%、180%、240% にスケーリングされたときにピクセル シフトが発生することがあります。

複数の画面サイズをサポートするためのガイドライン

Windows ストア アプリと Windows Phone ストア アプリは、さまざまな画面サイズや解像度のさまざまなデバイスで実行できますが、ビットマップの場合は拡大または縮小した場合に悪影響を受ける可能性があります。ユーザーも、画面の向きを変更できます。アプリが Windows で実行されている場合は、アプリのサイズを連続的に最小幅まで調整でき、また他のアプリと横に並べて表示できます。ユーザーがアプリを電話、タブレット、ノート PC、デスクトップ、PPI デバイスのいずれで実行する場合でも、UI が適切に表示され、機能を維持することを確認します。すべてのサイズと向きの画面に対応するように UI を設計する際は、次のガイドラインに従ってください。

Windows ストア アプリで狭いウィンドウ サイズに対応するための具体的なガイダンスについては、「幅の狭いレイアウトのガイドライン」を参照してください。

説明

多くの場合、アプリは電話の小さな画面から、中型ノート PC の画面、さらに場合によっては、オールインワン PC や PPI デバイスの画面にまで至るさまざまなサイズの画面で実行されます。画面サイズと解像度によって、アプリで利用できる表示可能領域の大きさが異なります。

次の用語は、さまざまな画面サイズに合わせたスケーリングを理解するうえで重要です。

用語	説明
画面サイズ	画面の物理的なサイズ (インチ単位)。通常は対角線の長さが計測されます。
画面解像度	画面でサポートされているピクセル数。1366x768 のように縦横のサイズで表されます。
縦横比	幅と高さの比率で示される画面の形状。16:9 などです。

プラットフォーム、コントロール、テンプレートは、いずれも、さまざまな画面サイズに対応できるように設計されています。 アアプリのレイアウトの多くは、ディスプレイを変更すると自動的に調整されますが、最上位レイアウト、コンテンツ領域、アプリ ナビゲーション、コマンドがすべての画面に想定どおり直感的に配置されるように、これらの要素について考慮が必要です。

次の表に、アプリを設計するときに考慮する最も重要な画面サイズに示します。

全画面の画面サイズ

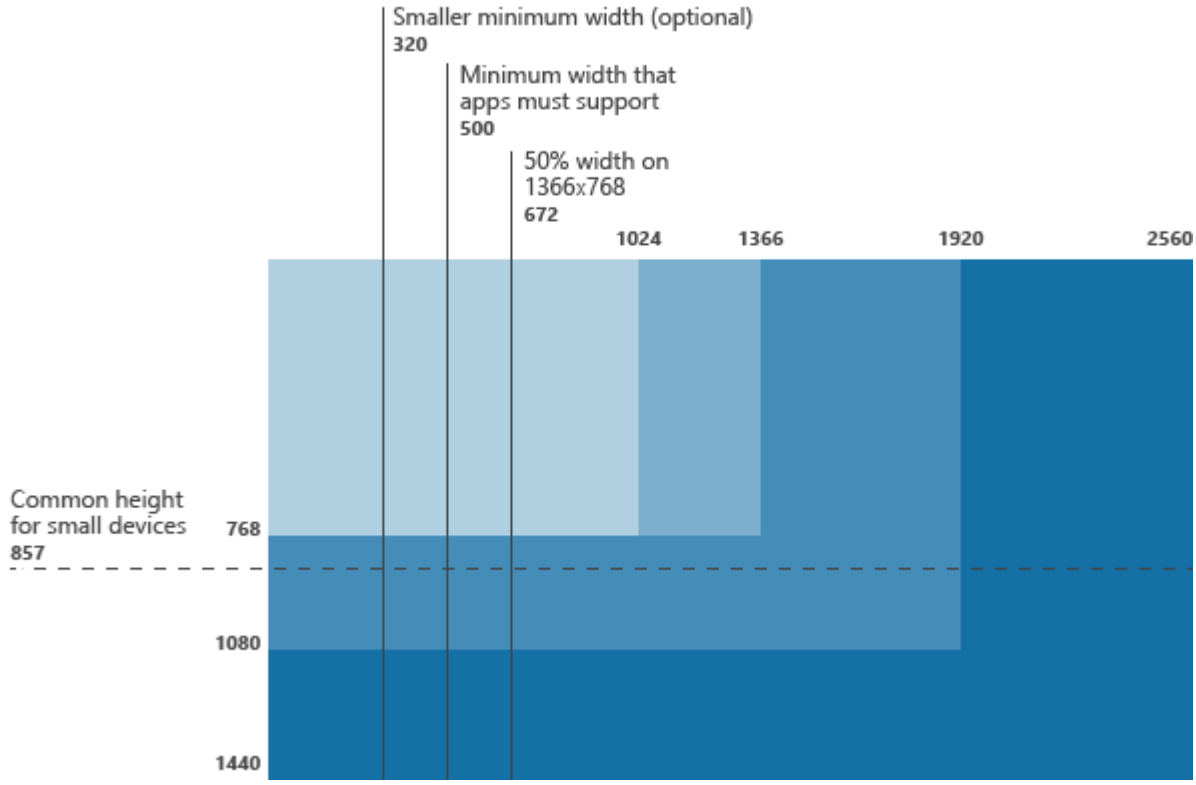
(有効ピクセル解像度)	**デバイスの説明**
1366x768	スレート、コンバーチブル PC、多くのノート PC (16:9 の縦横比)。ノート PC とデスクトップの基本解像度
1920x1080	大型のノート PC やデバイス (16:9 の縦横比)
2560x1440	非常に大型のオールインワン デバイス (16:9 の縦横比)
1280x800 と 800x1280	縦方向優先の小型デバイス (16:10 の縦横比)
1024x768 と 768x1024	横方向優先の小型デバイス (4:3 の縦横比)
1371x857 と 857x1371	小型デバイス (16:10 の縦横比)
384x640	4.5" 電話 (15:9 の縦横比)
400x711	4.7" 電話 (15:9 の縦横比)
450x800	55" 電話 (15:9 の縦横比)
491x873	6" 電話 (15:9 の縦横比)

Windows Phone ではなく Windows で動作するアプリを設計する際は、ユーザーが画面上に 2 つのアプリを同時に表示する場合や、アプリのサイズを最小幅に変更した場合に利用できる画面のサイズを考慮します。

分割画面サイズ

(有効ピクセル解像度)	**説明**
672x768	1366x768 デバイスで半分に分割した画面
500x768	アプリの既定の最小サイズ、1024x768 デバイスで半分に分割した画面
320x768	320 ピクセルの最小幅をサポートするアプリの最小サイズ

スケーリングに関する推奨事項について詳しくは、「ピクセル密度に合わせたスケーリングのガイドライン」をご覧ください。



推奨と非推奨

- 自動的に再配置されるコンテンツをサポートするには、可能であれば、可変コントロールを使います。可変コントロールには **XAML Grid control**、CSS グリッド、CSS 段組レイアウト、**ScrollViewer control** があります。たとえば、グリッド コントロールは、ディスプレイ デバイスの画面解像度に応じて、利用可能なスペースを埋めるように UI の特定のセクションのサイズを調整し、利用可能な画面スペースに応じてさまざまなセルにコンテンツを割り当てます。
- 最小サイズの画面に合わせて表示を調整し、機能を維持できるように、アプリのレイアウトとすべてのコントロールを設計します。
  - 既定の Windows ストア アプリの最小幅: 500px。
  - 既定以外の Windows ストア アプリの最小幅: 320px。
  - Windows Phone ストア アプリの最小サイズ (調整不可): 384px (縦)、640px (横)。

- UI とコントロールは、最小サイズ (上記を参照) までのすべての画面サイズで使用できる必要があります。考慮する必要がある重要なコントロールは、次のとおりです。
  - 上部と下部のアプリ バー
  - メッセージ ダイアログ (Windows ストア アプリ)
  - フライアウト (Windows ストア アプリ)
  - 設定ウィンドウ (Windows ストア アプリ)
  - ピボット コントロール (Windows Phone)
  - ハブ コントロール (Windows Phone)
- 大画面上の領域を効果的に使い、自動的に再配置されるレイアウトを持つようにアプリを設計します。大きな空白の領域を残さないようにします。
- アプリで最も重要なデバイス サイズで適切に動作するかテストします。実際のデバイスでのアプリのテストに加え、Windows ストア アプリ向けの Microsoft Visual Studio シミュレータを使って、さまざまな物理的な画面サイズ、解像度、向きでアプリの実行をシミュレートできます。
- すべての **input fields** の最小サイズを指定します。最小サイズを指定すると、ユーザーがウィンドウのサイズを変更しても入力フィールドが消えません。
- アプリの入力フィールドがソフト キーボードによって隠れないかどうかをテストします。
- 絶対配置を使わないように注意してください。使い方を誤ると、UI がウィンドウ サイズと向きの変更に対応しなくなる可能性があります。レイアウトをハード コーディングするのではなく、実行時に計算された位置を使って UI をレイアウトします。
- さまざまなピクセル密度で設計します。詳しくは、「ピクセル密度に合わせたスケーリングのガイドライン」をご覧ください。

Windows ストア アプリのみ

- アプリが既定の最小幅である 500 ピクセルのサイズまで機能することを確認します。具体的な推奨事項について詳しくは、「幅の狭いレイアウトのガイドライン」をご覧ください。

- 小さいサイズの方がアプリの動作が正確になる場合、ユーザーに対して画面にアプリを保持するように促すのであれば、既定以外の最小幅である 320 ピクセルをサポートすることもできます。
- アプリのサイズを変更してもユーザーが現在の作業を継続できることを確認します。たとえば、アプリの現在のページ、スクロール バーの状態、オプション、フォーカスを保持します。
- すべての画面サイズでチャームをサポートします。フライアウトとウィンドウが適切にスケーリングされることを確認します。

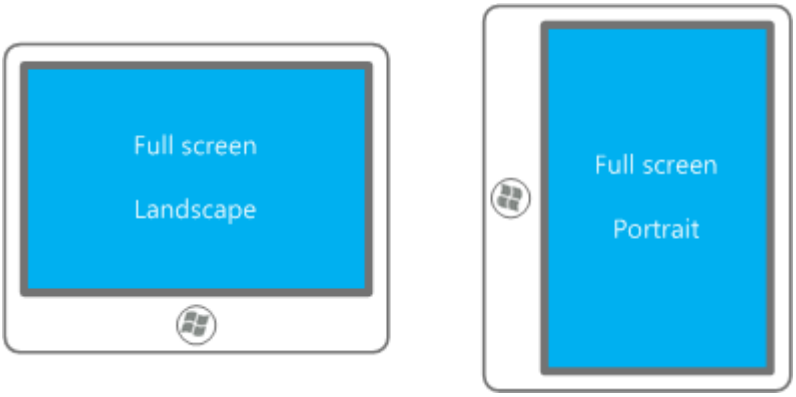
注 Windows 8 ではユーザーは、3 つの表示状態 (全画面、スナップ、ページ横幅) にのみアプリのサイズを変更できました。Windows 8.1 ではユーザーは、全画面から最小幅に至るまで、任意の幅にアプリのサイズを変更できます。

その他の使い方のガイダンス

向きの変更の自動的なサポート

ユーザーは電話、タブレット、モニターを回転させることができます。アプリが固定レイアウトを使っていない限り、Windows によって、横方向と縦方向の両方が自動的に処理されます。開発者は、アプリの幅がレイアウトに与える影響のみを検討する必要があります。

可変レイアウトを使うことが適切でないと考えられる場合について詳しくは、下の「固定レイアウト」のセクションをご覧ください。



サイズ変更された場合のアプリの動作 (Windows ストアのアプリのみ)

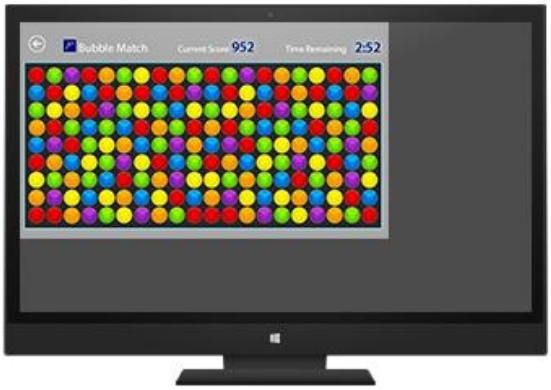
ユーザーが画面に複数のアプリを表示している場合、次の独特の UI 操作に注意してください。

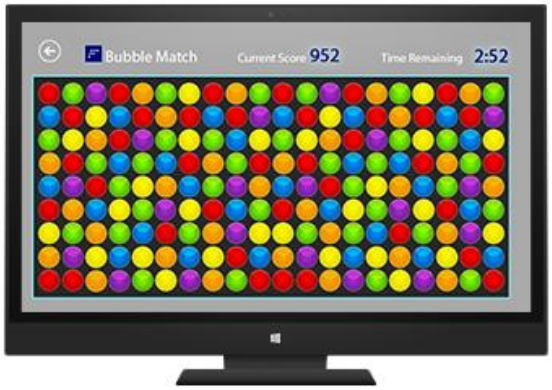
- ユーザーがチャームを呼び出した場合、チャームは、ユーザーが使った最後のアプリに適用されます。アプリのサイズや画面上の位置は関係ありません。
- 画面上の各アプリの間にはハンドルがあります。ユーザーは、ハンドルをスライドすることによってアプリ ウィンドウのサイズを変更します。ハンドルには、どのアプリにフォーカスがあるかも示されます。
- ユーザーがアプリ間のハンドルをつかんで、アプリの最小幅未満の幅にアプリのサイズを変更しようとすると、アプリは画面に表示されなくなります。
- 複数のアプリが画面に表示されているときに、ユーザーがデバイスやモニターを回転させた場合、アプリの向きは切り替わりません。

固定レイアウト

ほとんどのアプリでは、画面のサイズと解像度の変化に対応してコンテンツが自動的に再配置される、動的レイアウトを使うことができます。ただし、場合によっては、固定レイアウトが必要になります。ゲームのように、コンテンツのみが正しくても成立せず、グラフィックの正確な表示に依存するアプリでは、固定 (絶対) レイアウトを使う必要があります。Windows では、プラットフォームに組み込まれている "サイズに合わせたスケーリング" というアプローチでこれらのアプリに対応します。

さまざまな画面サイズに対応できない固定レイアウトがアプリに必要と判断した場合は、サイズに合わせたスケーリングのアプローチを使って、さまざまなサイズの画面全体に固定レイアウトが表示されるようにします。次の図をご覧ください。



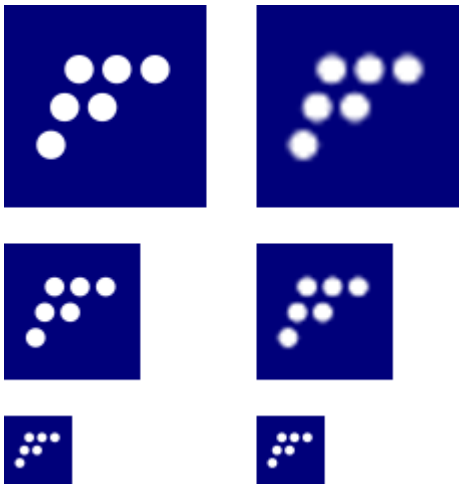


サイズに合わせたスケーリングを実装するには、次の操作を実行します。

- 基本解像度 (1366 x 768 などや 384x640 ピクセル (電話) など)) のレイアウトを設計します。これは、大きな画面に合わせてスケーリングされるレイアウトです。
- 固定コンテンツは ViewBox コントロール内に配置します (「ViewBox in JavaScript and HTML」または「Viewbox in C#/VB/C++ and XAML」)。ViewBox コントロールは、固定レイアウトが画面に収まるようにスケーリングします。
- ViewBox コントロールの幅と高さが 100% になるようにします。
- ViewBox の固定サイズ プロパティをレイアウトの固定ピクセル サイズに合わせて定義します (1366 x 768、384x640 などなど)。
- レター ボックスの色を選びます。固定コントロールは、縦横比や画面サイズの変化に合わせて動的に変更されません。そのため、サイズに合わせたスケーリングの手法により、アプリのコンテンツが自動的に中央に配置され、レター ボックス化されます (横方向または縦方向)。レター ボックス バーの色は、最上位アプリ レイアウトの色によって決まります。ハードウェアと調和する黒などの暗色、意図が感じられる灰色などの中間色、アプリ コンテンツの色と調和する色を選ぶことをお勧めします。
- ベクター アセットまたは高解像度アセットを用意するサイズに合わせたスケーリングの手法では、アプリケーションは大画面のデスクトップ モニター上でアプリの設計サイズの最大 180% (Windows の場合) または 280% (Windows Phone の場合) のサイズまでスケーリングされます。スケーラブル ベクター グラフィックス (SVG)、Extensible Application Markup Language (XAML)、デザイン プリ

ミティブなどのベクター アセットは、スケーリング アーティファクトが発生したり、ぼやけたりすることなくスケーリングされます。ラスター アセット (ビットマップ イメージなど) が必要な場合は、MRT アセットを用意します。

次の図に、スケール アップしたときに、スカラー イメージ (右) がベクター イメージ (左) と比べてどの程度劣化するかを示します。



- アダプティブ コントロールは ViewBox コントロールに配置しないでください。

スケーリングに関するその他の推奨事項について詳しくは、「ピクセル密度に合わせたスケーリングのガイドライン」を参照してください。

アニメーション

目的がはっきりし、適切にデザインされたアニメーションは、アプリを生き生きとさせ、精巧で洗練された印象を与えます。コンテキストの変化がわかりやすく、視覚的な切り替えがエクスペリエンスに結び付きます。

このセクションの内容

トピック	説明
追加と削除	リスト アニメーションを使うと、写真のアルバムや検索結果の一覧などのコレクションに対して任意の数の項目を挿入または削除できます。
コンテンツ切り替え	コンテンツ切り替えアニメーションを使うと、コンテナーや背景はそのままに、画面のある領域のコンテンツを変更できます。新しいコンテンツはフェード インします。既にあるコンテンツを差し替える場合、そのコンテンツはフェード アウトします。
ドラッグ	ドラッグ アンド ドロップ アニメーションは、リスト内で項目を移動するときや、特定の項目を別の項目上にドロップするときなど、オブジェクトを移動する際に使います。
エッジに基づく UI アニメーション	エッジに基づく UI アニメーションでは、画面の端を起点とする UI の表示と非表示を切り替えられます。
フェード	フェード アニメーションは、項目を画面に表示したり、項目を画面から非表示にするときに使います。フェード アニメーションには、フェード イン、フェード アウト、クロスフェードの 3 種類があります。
ページ切り替え	ページ切り替えアニメーションを使うと、新たに起動したアプリの最初のページを表示したり、アプリ内でページを切り替えたりできます。

ポインター クリック	ポインター アニメーションを使って、項目のタップまたはクリックに対する視覚的なフィードバックをユーザーに提供します。ユーザーが項目をタップまたはクリックすると、ポインター ダウン アニメーションが再生されます。このアニメーションでは、項目が若干縮小され、押されていることを示します。クリックまたはタップが解放されると、ポインター アップ アニメーションが再生されます。このアニメーションでは、項目が元のサイズに戻り、解放されたことを示します。
位置変更	位置の変更アニメーションを使って、1 つまたは複数の要素を新しい位置に移動します。
ポップアップ	ポップアップ アニメーションを使って、コンテキスト メニューやフライアウトなど、ポップアップ UI の表示と非表示を切り替えます。ポップアップ要素とは、アプリのコンテンツの上に表示されるコンテナーのことで、ユーザーがポップアップ要素の外部をタップまたはクリックすると消えます。
スワイプ	スワイプ アニメーションは、項目の選択にスワイプ ジェスチャを実装する場合に使います。

追加と削除のアニメーションのガイドライン

リスト アニメーションを使うと、写真のアルバムや検索結果の一覧などのコレクションに対して任意の数の項目を挿入または削除できます。

特に指定がない限り、これらのガイドラインはリスト アニメーションと検索リスト アニメーションのどちらにも適用されます。

推奨と非推奨

- リスト アニメーションは、既にある一連の項目に新しい項目を 1 つ追加するときに使います。たとえば、新しい電子メールを受け取ったときや、既にあるセットに新しい写真をインポートするときに使います。
- リスト アニメーションは、一連の項目に対して複数の項目を一度に追加するときに使います。たとえば、一連の新しい写真を既にあるコレクションにインポートするときに使います。複数項目の追加と削除は、個々のオブジェクトの処理に間が生じることなく同時に実行されます。
- 追加と削除のリスト アニメーションは、ペアで使います。一方のアニメーションを使った場合は、逆の操作として対応するもう一方のアニメーションを使うようにしてください。
- リスト アニメーションは、要素または要素のグループを一度に追加または削除できる項目リストに使います。
- 検索リスト アニメーションは、検索フィルターにワード ホイールを使っているなど検索結果が次々に再配置される場合に使います。
- コンテナーを表示したり非表示にしたりする目的でリスト アニメーションを使うのは避けてください。リスト アニメーションは、既に表示されているコレクションまたはセットのメンバーに対して使います。アプリ サーフェス上に一時的なコンテナーを表示したり非表示にしたりするには、フライアウト アニメーションを使います。アプリ サーフェスの一部となっているコンテナーを表示したり置き換えたりするには、コンテンツ切り替えアニメーションを使います。
- 項目のセット全体に対してリスト アニメーションを使うのは避けてください。コンテナー内のコレクション全体を追加したり削除したりするには、コンテンツ切り替えアニメーションを使います。

コンテンツ切り替えのアニメーションのガイドライン

コンテンツ切り替えアニメーションを使うと、コンテナーや背景はそのままに、画面のある領域のコンテンツを変更できます。新しいコンテンツはフェード インします。既にあるコンテンツを差し替える場合、そのコンテンツはフェード アウトします。

推奨と非推奨

- コンテンツ切り替えは、空のコンテナーに一連の新しい項目を流し込むときに使います。たとえば、アプリの初期読み込みの直後は、アプリのコンテンツの一部が表示に間に合わない場合があります。このような場合、コンテンツを表示する準備が整った段階で、コンテンツ切り替えアニメーションを使い、遅れてコンテンツが表示されるようにします。
- あるコンテンツの組み合わせを、画面内の同じコンテナー内に既に存在する別のコンテンツの組み合わせに置き換えるときに、コンテンツ切り替えを使います。
- 新しいコンテンツを画面に表示するときには、一般的なページのフロー (読む方向) とは逆の方向にコンテンツをスライドさせます。たとえば、左から右に読むドキュメントに対して新しいコンテンツを流し込むアニメーションの場合、新しいコンテンツは右から左に流し込むのが自然です。
- 新しいコンテンツは論理的な流れで配置します。たとえば、最も重要なコンテンツを最後にします。
- 更新対象のコンテンツを含んだコンテナーが複数存在する場合、切り替え効果アニメーションは、間を置かずにすべて同時にトリガーします。
- コンテンツ切り替えアニメーションは、ページの全体が変化する場合には使わないでください。この場合には、ページ切り替えアニメーションを使います。
- コンテンツの更新のみであれば、コンテンツ切り替えアニメーションは使わないでください。コンテンツ切り替えアニメーションは、動きを表現するために使います。更新には、フェード アニメーションを使ってください。

ドラッグ アニメーションのガイドライン

ドラッグ アンド ドロップ アニメーションは、リスト内で項目を移動するときや、特定の項目を別の項目上にドロップするときなど、オブジェクトを移動する際に使います。

推奨と非推奨

ドラッグ開始アニメーション

- ドラッグの開始アニメーションは、ユーザーがオブジェクトを動かし始めるときに使います。
- ドラッグ アンド ドロップ操作の影響を受けるオブジェクトが他に存在する場合に限り、それらのオブジェクトをアニメーションに含めるようにします。
- ドラッグの開始アニメーションがトリガーされるまでの間には、ユーザーがある程度オブジェクトを動かせるようにします。単にタップまたは選択しようとしたオブジェクトがユーザーの意図に反してドラッグされるのを防ぐことができます。しきい値は 20 TIP (タッチに依存しないピクセル) がお勧めです。
- ドラッグの開始アニメーションによって始まったアニメーションのシーケンスの終了には、ドラッグの終了アニメーションを使います。ドラッグの終了アニメーションにより、ドラッグの開始アニメーションで変化したドラッグされたオブジェクトのサイズが元に戻ります。

ドラッグの終了アニメーション

- ドラッグの終了アニメーションは、ドラッグされたオブジェクトをドロップするときに使います。
- ユーザーがオブジェクトをドロップしてリストの並びを変えるときは通常、他の項目の位置を変更して、項目をドロップするためのスペースを確保する必要があります。これには、ドラッグの終了アニメーションの終了後に、項目をリストへの追加アニメーションを呼び出します。ただし、項目は追加しません。追加する項目は、既に存在しています。これにより、すべての要素にアニメーションが実行され、それぞれが適切な位置に移動します。

- ドロップ後にドラッグ ソースを消す場合 (ファイルをフォルダー アイコンの上にドロップしてフォルダーにファイルを格納する場合など)、ドラッグ ソースにフェード アウト アニメーションを使います。
- ドラッグの開始アニメーションに影響を受けるオブジェクトが存在する場合に限り、それらのオブジェクトをドラッグの終了アニメーションに含めるようにします。
- ドラッグの終了アニメーションは、ドラッグの開始アニメーションよりも先に使わないでください。ドラッグ シーケンスの完了後にオブジェクトを元のサイズに戻すためには、両方のアニメーションを使う必要があります。

項目間でのドラッグの開始アニメーション

- 項目間へのドラッグの開始アニメーションは、2 つのオブジェクトの間のドロップ可能な場所にドラッグ ソースをドラッグするときに使います。
- 適度な大きさのドロップ ターゲット領域を選んでください。この領域が小さすぎると、ドラッグ ソースをドロップする際に重ね合わせるのが難しくなるため、好ましくありません。
- ドロップ可能な場所を示すために影響を受けるオブジェクトが移動する距離は、40 ピクセルをお勧めします。
- ドロップ可能な場所を示すために影響を受けるオブジェクトが移動するときには、互いにまっすぐに引き離すことをお勧めします。移動方向が上下になるか、左右になるかは、影響を受けるオブジェクトが並ぶ向きによって異なります。
- ドラッグ ソースを領域内にドロップできない場合、項目間でのドラッグの開始アニメーションは使わないでください。項目間へのドラッグの開始アニメーションは、影響を受けるオブジェクトの間にドラッグ ソースをドラッグできることをユーザーに知らせるためのものです。

項目間でのドラッグの中止アニメーション

- 項目間へのドラッグの中止アニメーションは、ユーザーがオブジェクトをドラッグして 2 つのオブジェクトの間のドロップ可能な領域から出すときに使います。
- 項目間でのドラッグの開始アニメーションよりも先に、項目間でのドラッグの中止アニメーションを使わないでください。

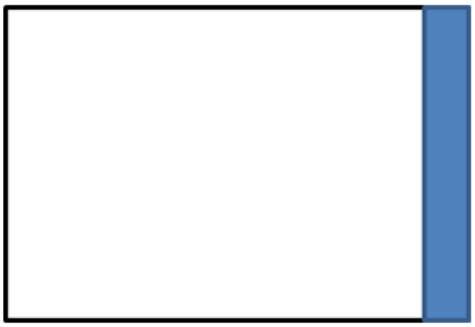
エッジに基づく UI アニメーションのガイドライン

エッジに基づく UI のアニメーションでは、画面のエッジ (端) を起点とする UI の表示と非表示を切り替えられます。 この表示と非表示のアクションは、ユーザーが開始することも、アプリから開始することもできます。UI は、アプリの手前に表示するか、メイン アプリ サーフェスの一部として表示することができます。UI をアプリ サーフェスの一部として表示する場合は、UI を表示できるようにアプリの残りの部分のサイズを調整する必要があります。

推奨と非推奨

- 画面領域をあまり占有しないカスタム メッセージ バーやエラー バーを表示または非表示にするには、エッジ (端) UI アニメーションを使います。
- 作業ウィンドウやカスタム ソフト キーボードなど、画面内側にスライドして領域を大きく確保する UI を表示するには、パネル アニメーションを使います。
- UI を開くには、それが関連付けられているエッジ (端) から画面内側にスライドします。
- UI を閉じるには、画面内側から、開いたときと同じ端に向かってスライドします。
- UI のスライド操作に応じてアプリのコンテンツ サイズを変更する必要がある場合は、フェード アニメーションを使ってサイズを変更します。
    - UI を画面内側に向かってスライドする場合は、エッジ (端) UI アニメーションまたはパネル アニメーションの後にフェード アニメーションを使います。
    - UI を画面外側に向かってスライドする場合は、エッジ (端) UI アニメーションまたはパネル アニメーションと同時にフェード アニメーションを使います。

- 通知には、このアニメーションを適用しないでください。エッジに基づく UI に通知を格納することはお勧めしません。
- 画面のエッジ (端)にない UI コンテナーやコントロールには、エッジ (端) UI アニメーションとパネル アニメーションを適用しないでください。このアニメーションは、画面のエッジ (端)にある UI の開閉とサイズ変更にのみ使います。他のタイプの UI を移動するには、位置変更アニメーションを使います。



Use edge UI animations

Use reposition animation

フェード アニメーションのガイドライン

フェード アニメーションは、項目を画面に表示したり、項目を画面から非表示にするときに使います。フェード アニメーションには、フェード イン、フェード アウト、クロスフェードの 3 種類があります。

推奨と非推奨

- アプリで互いに関係のない要素や、テキストの多い要素を切り替えるときには、クロスフェードではなくフェード アウトとフェード インを使います。そうすることで、差し替え前のオブジェクトが完全に消えてから差し替え後のオブジェクトを表示させることができます。クロスフェードは、テキストが多い場合には特にお勧めしません。
- クロスフェード アニメーションは、サイズの異なる要素を切り替えるときや、大きな領域を更新するときに使います。アニメーションの途中には、差し替え前の要素、差し替え後の要素とも半透明になり、背景が見えるようになります。
  Extensible Application Markup Language (XAML) と共に使うプログラミング言語には、専用のクロスフェード アニメーションがありません。そのような言語では、フェード インとフェード アウトのアニメーションをタイミングが重なるように使えば、クロスフェードと同じ効果を実現できます。
- 差し替える 2 つの要素のサイズが一定であり、ユーザーに同じ項目を見ているような印象を与えたいときには、差し替え後の要素を差し替え前の要素の上にフェード インさせます。差し替え後の項目を差し替え前の項目の上にフェード インできます。フェード インが完了したら、差し替え前の項目は消すことができます。これはもちろん、差し替え後の項目が差し替え前の項目を完全に覆い隠せる場合にのみ可能な方法です。この方法にはこのほか、切り替え中に背景が見えることを防ぐ効果があります。
- リストの項目を追加または削除する目的でフェード アニメーションを使うのは避けてください。そのような目的には、専用のリスト アニメーションを使います。
- フェード アニメーションは、ページの全コンテンツを変化させるときには使わないようにしてください。そのような目的には、専用のページ切り替えアニメーションを使います。

ページ切り替えアニメーションのガイドライン

ページ切り替えアニメーションを使うと、新たに起動したアプリの最初のページを表示したり、アプリ内でページを切り替えたりできます。

注 スクリーンの一部の領域を占めるだけのコンテンツを切り替えるときには、ページ切り替えアニメーションではなく、コンテンツ切り替えアニメーションを使います。

推奨と非推奨

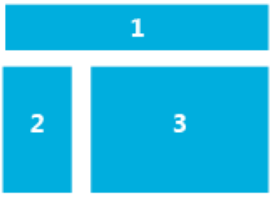
- ページは、自然な境界に沿って 2 ～ 5 の領域セットに分割してください。各領域に適用するタイミングをずらすと、領域が一度に全部ではなく、順番に表示されます。また、幅が広いアプリのレイアウトでは、領域が 100 ピクセルずつオフセットされます。アプリに幅が狭い状態用の特別なレイアウトを使う場合は、小さいオフセットを使うことができます。
- 差し替え前のページと差し替え後のページに共通するコンテンツがそのままの位置にとどまり、そのコンテンツにアニメーションが適用されていないことを確認します。たとえば、差し替え前のページと差し替え後のページのどちらにも **[戻る]** ボタンがある場合には、そのボタンは切り替えアニメーションの対象にしません。
- 差し替え前のページに **[戻る]** ボタンがなく (アプリの最初のページなど)、差し替え後のページにはあるという場合には、その **[戻る]** ボタンは別の領域に指定し、アニメーションを適用して他の領域よりも先に表示します。
- 差し替え前のページと差し替え後のページとの間で背景が異なる場合には、フェード アニメーションを使って新しい背景を表示します。ページ切り替えアニメーションと同時にフェード イン アニメーションを開始します。
- 差し替え後のページの一部のコンテンツが表示にすぐに表示できる状態にない場合には、ページ切り替えアニメーションを使うとその時点で準備ができているコンテンツが流し込まれます。その間に必要があれば、残りのコンテンツを準備している間に [プログレス コントロール] を表示します。残りのコンテンツの表示の準備が整ったら、コンテンツ領域に基づいてアニメーションを使って流し込みます。コンテンツ領域が大きい場合には、コンテンツ切り替えアニメーションを使います。コンテンツ領域が小さい場合や、連続性のないコンテンツの場合には、フェード イン アニメーションを使います。

- 画面に表示するページは、一般的なページのフロー (読む方向) とは逆の方向にスライドさせます。たとえば、差し替え後のページのコンテンツを読む方向が左から右の場合、新しいページは右から左にスライドするのが自然です。右から左に読むアプリの場合、新しいページは左から右にスライドするのが自然です。同じように、下の図に示すように 1 つのページを分割する場合、分割した部分を表示する順番は、読む方向と逆にするのが自然です。
- ユーザーがアプリ ウィンドウのサイズを変更するときには、ページ切り替えアニメーションを実行しないでください。ページ切り替えアニメーションは、特定のビューを使っているときにページ間で移動する場合に限られます。ビューが変わるときには、システムがレイアウトの切り替えのアニメーションを処理します。

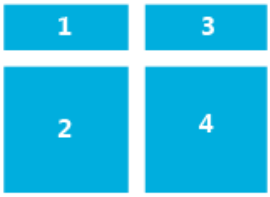
その他の使い方のガイダンス

ここでは、最もよくみられるページの分割方法について、表示の順番も含めて説明します。

2 stages: Header, content

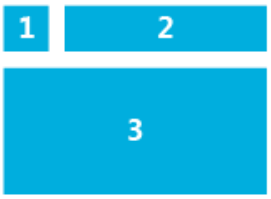


3 stages: Header, left content, right content

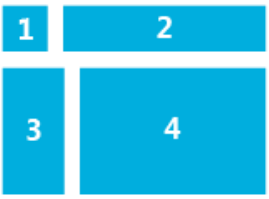


4 stages: Left header, left content, right header, right content

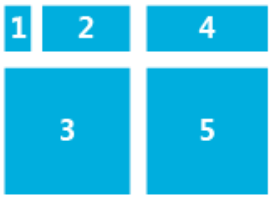
[戻る] ボタンのある一般的なページ分割を次に示します。差し替え前のページに **[戻る]** ボタンがなく (アプリの最初のページなど)、差し替え後のページにはあるという場合には、その **[戻る]** ボタンは別の領域に指定し、アニメーションを適用して他の領域よりも先に表示します。



3 stages: Back button, header, content



4 stages: Back button, header, left content, right content

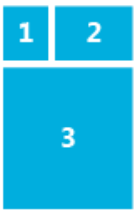


5 stages: Back button, left header, left content, right header, right content

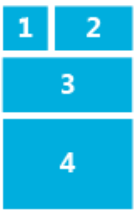
ここでは、狭い幅のビューまたは縦向きビューで表示されるアプリに使うページ分割方法のうち最もよく見られるものについて、分割の表示の順番も含めて説明します。差し替え前のページに既に **[戻る]** ボタンがある場合、そのボタンは切り替えアニメーションの対象にせず、そのままの位置を維持します。

2 stages: Header, content

3 stages: Header, top content, bottom content



3 stages: Back button, header, content



4 stages: Back button, header, top content, bottom content

ポインター クリック アニメーションのガイドライン

ポインター アニメーションを使って、項目のタップまたはクリックに対する視覚的なフィードバックをユーザーに提供します。ユーザーが項目をタップまたはクリックすると、ポインター ダウン アニメーションが再生されます。このアニメーションでは、項目が若干縮小され、押されていることを示します。クリックまたはタップが解放されると、ポインター アップ アニメーションが再生されます。このアニメーションでは、項目が元のサイズに戻り、解放されたことを示します。

推奨と非推奨

- ポインター アップ アニメーションを使うときには、タップまたはクリックして指を離した直後にアニメーションを開始するようにします。ポインター アップ アニメーションがすぐに表示されることが重要です。これにより、タップまたはクリックによってトリガーされたアクション (新しいページへの移動など) の応答が遅れたとしても、ユーザーの操作が認識されたというフィードバックを即座に返すことができます。

位置変更 アニメーションのガイドライン

位置の変更アニメーションを使って、1 つまたは複数の要素を新しい位置に移動します。

推奨と非推奨

- 位置変更アニメーションは、エッジに基づく UI を表示したり非表示にしたりするときには使わないでください。エッジに基づく UI とは、画面のエッジ （端）の 1 つに固定された要素またはコンテナーを指します。そのような場合には、専用のエッジに基づく UI アニメーションを使います。

ポップアップ UI アニメーションのガイドライン

ポップアップ アニメーションを使って、コンテキスト メニューやフライアウトなど、ポップアップ UI の表示と非表示を切り替えます。ポップアップ要素とは、アプリのコンテンツの上に表示されるコンテナーのことで、ユーザーがポップアップ要素の外部をタップまたはクリックすると消えます。

ポップアップ アニメーションの適切な使用

- ポップアップ アニメーションは、アプリのページに含まれない、コンテキスト メニュー、フライアウト、その他のコンテキスト対応 UI などのカスタム ポップアップ UI 要素を表示または非表示にするときに使います。ポップアップ要素が表示されるときに移動する距離は、50 ピクセルをお勧めします。 Windows で用意されているコモン コントロールには、既にこのアニメーションが組み込まれています。
- ツールチップやダイアログにポップアップ アニメーションを使わないでください。カスタム ツールチップやダイアログの表示や非表示にはフェード アニメーションを使います。
- アプリのメイン コンテンツの UI を表示または非表示にするときにはポップアップ アニメーションを使わないでください。メインのアプリのコンテンツの上に表示するポップアップ コンテナーを表示したり非表示にしたりする場合に限り、ポップアップ アニメーションを使います。

スワイプ アニメーションのガイドライン

スワイプ アニメーションは、項目の選択にスワイプ ジェスチャを実装する場合に使います。

推奨と非推奨

- スワイプ選択アニメーションは、ユーザーが項目をドラッグしたものの、その項目を離すまでに選択状態を切り替えるほどの移動がなかった場合に、選択状態を変えることなくその項目を元の位置に戻すときに使います。
- スワイプ アニメーションの方向は、ユーザーのスワイプ ジェスチャーの方向と一致するようにします。
  - 横方向にスクロールするコンテンツの場合には、スワイプ アニメーション シーケンスが下方向へ動いてから上に戻るよう、スワイプの方向を縦にします。
  - 縦方向にスクロールするコンテンツの場合には、スワイプ アニメーション シーケンスがアプリの読む方向に従って横に動くよう、スワイプの方向を横にします。
    - 左から右に読むアプリの場合、アニメーションは右に動いた後で左に動くのが自然です。
    - 右から左に読むアプリの場合、アニメーションは左に動いた後で右に動くのが自然です。
- スワイプ アニメーションの一環として項目を移動する距離は、縦方向に動かす場合には 15 ピクセル、横方向に動かす場合には 23 ピクセルがお勧めです。

コントロール

このセクションでは、利用可能なすべてのコントロールの設計に関する情報を、すばやく簡単に参照できるように 1 か所にまとめています。ここでは、カスタマイズせずに利用できる標準コントロールについて説明しています。また、実際のアプリでの各コントロールの使用例も示します。

注　ここで紹介するすべてのコントロールは、Adobe Photoshop および Adobe Illustrator 用のテンプレートに含まれています。テンプレートの入手方法については、ダウンロードに関するページをご覧ください。

標準コントロールを使うと、一般的な対話的操作を作成する際に時間を短縮できます。標準コントロールは、そのまま利用できます。開発者と共同で作業する場合は、設計でこれらのコントロールを参照することによって認識を共有できます。コントロールに関する開発者向けの情報については、「コントロールの一覧 (JavaScript と HTML を使った Windows ストア アプリ)」または「コントロールの一覧 (C#/VB/C++ と XAML を使った Windows ストア アプリ)」をご覧ください。

標準コントロールは単なる出発点にでしかありません。対話的操作のカスタマイズは不要であると考える人もいますが、独自の UI でユーザーを喜ばせ、アプリにブランド要素を組み込むことはビジネス チャンスであると考える人もいます。たとえば、あるアプリでは変更を加えないボタン コントロールが適切で手軽なソリューションとなり、別のアプリではカスタマイズしたボタンがアプリのブランド化された要素となります。

このセクションの内容

トピック	説明
アプリ バー	アプリ バーを使うと、ユーザーは必要なコマンドに簡単にアクセスできます。ユーザーが画面の上端または下端をスワイプするとアプリ バーが表示され、そのコンテンツを操作するとアプリ バーが消えます。アプリ バーは、ユーザーのコンテキストに固有のコマンドまたはオプション (写真の選択や描画モードなど) を表示するためにも使うことができます。アプリ バーは、アプリのページまたはセクションのナビゲーションにも使用できます。
戻るボタン	戻るボタンは、戻るナビゲーションをボタンの形式で提供します。

ボタン	ボタン (コマンド ボタン) は、特定の操作を直ちに実行する方法をユーザーに与えます。
チェック ボックス	アプリにチェック ボックスを追加する場合は、次のガイドラインに従ってください。
コンテキスト メニュー	コンテキスト メニューには、ユーザーが現在のコンテキストで使うことのできるコマンドとオプションの一覧を表示します。
DatePicker	DatePicker は、ユーザーがタッチ、マウス、またはキーボード入力を使ってローカライズされた日付を選択できる標準化された方法です。
ドロップ ダウンリスト	ドロップダウン リストを使うと、ユーザーは相互排他的な値の一覧からオプションを選ぶことができます。
FlipView	
フライアウト	フライアウト (flyout) は、ユーザーが現在操作している内容に関係する UI を一時的に表示するために使われる軽量なフライアウトです。
Windows アプリ ハブ	ハブ コントロールは、階層型ナビゲーション パターンを使って、リレーショナル情報アーキテクチャを使ったアプリをサポートします。
Windows Phone アプリ ハブ	ハブ コントロールには、左右に動かすことができる一連のセクションが表示されます。これは、アプリの全画面コンテナーであり、ナビゲーション モデルでもあります。
ラベル	ラベルは、コントロールまたは関連するコントロールのグループの名前やタイトルです。ラベルには機能は実装されません。
リンク	Windows ストア アプリにリンクを追加するためのガイドライン。

リスト ボックス (または選択)	リスト ボックス (選択とも呼ばれます) では、ユーザーが通常は 1 つ、ただし時に複数の項目を項目のリストから選択する手段が提供されます。リスト ボックスの項目は、すべての項目を表示する領域がない場合には、スクロールできます。
リスト ビュー と グリッド ビュー	グリッド ビューまたはリスト ビューは、Windows ストア アプリ内のコンテンツのコレクションです。
マップ	マップ コントロールでは、地図および上空からの写真、方向、検索結果、トラフィックを表示できます。
メッセージ ダイアログ	メッセージ ダイアログは、常にモーダルで明示的に閉じられる安定した状況依存のサーフェスを提供する、オーバーレイ UI 要素です。
ピボット	ピボット コントロールは、通常は同じデータ セット内の異なるピボット (ビューまたはフィルター) 間で、迅速な移動手段を提供する全画面表示のコンテナーおよびナビゲーション モデルです。たとえば、ピボット コントロールを使った電子メール アプリでは、最初のピボット項目 (またはビュー) 内のすべての電子メールを一覧に示した後、他のピボット項目で同じ一覧を未読の電子メール、フラグ付き電子メール、緊急の電子メールにフィルターできます。
プログレス	プログレス コントロールは、時間のかかる操作が進行中であることを示すフィードバックをユーザーに返します。
ラジオ ボタン	ラジオ ボタンでは、ユーザーは 2 つ以上の選択肢から 1 つのオプションを選ぶことができます。
評価	評価コントロールは、評価を示すアイコンをクリックすることで、ユーザーが何かを評価できるようにします。評価には、平均評価、暫定評価、ユーザー評価の 3 種類があります。
検索	アプリのコンテンツに検索結果を提供するには、次のガイドラインに従ってください。このガイドラインには、検索ボックスで検索候補とプレースホルダー テキストを提供する場合や検索結果ページを設計する場合のヘルプを記載しています。
スクロール バー	パンとスクロールを行うと、画面の境界外のコンテンツを拡張表示することができます。

セマンティック ズーム	セマンティック ズーム コントロールを使うと、ユーザーは同じデータ セットの 2 つの異なるセマンティック表示間でズームを実行できるようになります。
スライダー	Windows ストア アプリにスライダーを追加するには、次のガイドラインに従ってください。
TimePicker	TimePicker は、ユーザーがタッチ、マウス、またはキーボード入力を使ってローカライズされた時刻を選ぶことができる標準化された方法です。
トグル スイッチ	トグル スイッチは、ユーザーが項目をオンまたはオフに切り替えることができる物理的なスイッチを模したものです。Windows ストア アプリにトグル スイッチ コントロールを追加するには、次のガイドラインに従ってください。
ツールチップ	ユーザーに操作の実行を指示する前に、ツールチップを使ってコントロールに関する詳しい情報を表示します。
Web ビュー	Web ビュー コントロールは、Internet Explorer のように動作するビューをアプリに組み込みます。また Web ビュー コントロールでは、ハイパーリンクの表示と動作が可能です。

アプリ バーのガイドライン

説明

アプリ バーを使うと、ユーザーは必要なコマンドに簡単にアクセスできます。ユーザーが画面のエッジ (上端、または下端)をスワイプするとアプリ バーが表示され、そのコンテンツを操作するとアプリ バーが消えます。アプリ バーは、ユーザーのコンテキストに固有のコマンドまたはオプション (写真の選択や描画モードなど) を表示するためにも使うことができます。アプリ バーは、アプリのページまたはセクションのナビゲーションにも使用できます。

ユーザーがワークフロー (製品の購入など) を実行するために必要なコマンドがある場合は、それらのコマンドをアプリ バーではなくキャンバスに配置します。

上部のアプリ バー

ナビゲーション バーまたは上部のアプリ バーは、ユーザーがアプリの別の領域にアクセスできるようにするためのナビゲーション コントロールを配置するのにお勧めの場所です。ナビゲーションに上部のアプリ バーを使うことで、Windows ストア アプリのナビゲーションで一貫性のある予測可能なユーザー エクスペリエンスが実現します。一貫性があることで、ユーザーは安心してシステム内を移動でき、アプリのナビゲーションに関する知識をさまざまなアプリで利用することができます。

ユーザーがキャンバスだけで主なシナリオを完了できる必要があります。ナビゲーション バーは、ナビゲーション コントロールの 2 次的な場所です。ナビゲーション バーを使うと、アプリのあらゆる部分にユーザーを導くことができ、ホーム ページにすばやくアクセスできるようになります。また、アプリのさまざまな部分にジャンプして探索するようにユーザーを促すことができます。

新しいものを作るための "+" ボタンの追加や検索ボックスの統合など、上部のアプリ バー内にその他の機能を追加することもできます。その他の機能を追加する場合は、アプリ バーの右側にそれらを配置することをお勧めします。

ナビゲーション バーの項目のスタイルは指定できますが、既定の外観は単純なボタンです。ただし、先の画像で示されているように、ボタンとサムネイルを使うという表現方法も一般的です。

ストア アプリのように、複数の領域にナビゲーション バーを分割することもできます。ご覧のように、上部のセクションはグローバル ナビゲーション用であり、下部のセクションはアプリ カテゴリ用です。

下部のアプリ バー

下部のアプリ バーは、コマンドを配置するのにお勧めの場所です。コマンドをアプリのキャンバスからアプリ バーに移動すると、ユーザーにとって最もイマーシブな操作性が実現します。

標準的なアプリ バー コントロールは、カスタム作業を (ほとんど) 伴わずにアプリ バーを実装したい開発者を対象としています。アプリ バーを作ることは簡単ですが、アプリ バーが Windows のガイダンスとパターンに従って動作するようにすることは簡単ではありません。**CommandBar class** と **WinJS.UI.AppBar object** は意図された設計と動作に適応するようになっているため、開発者は細かいことを考える必要がなく、一般的なコマンド実行パターンを外れる可能性は低くなります。

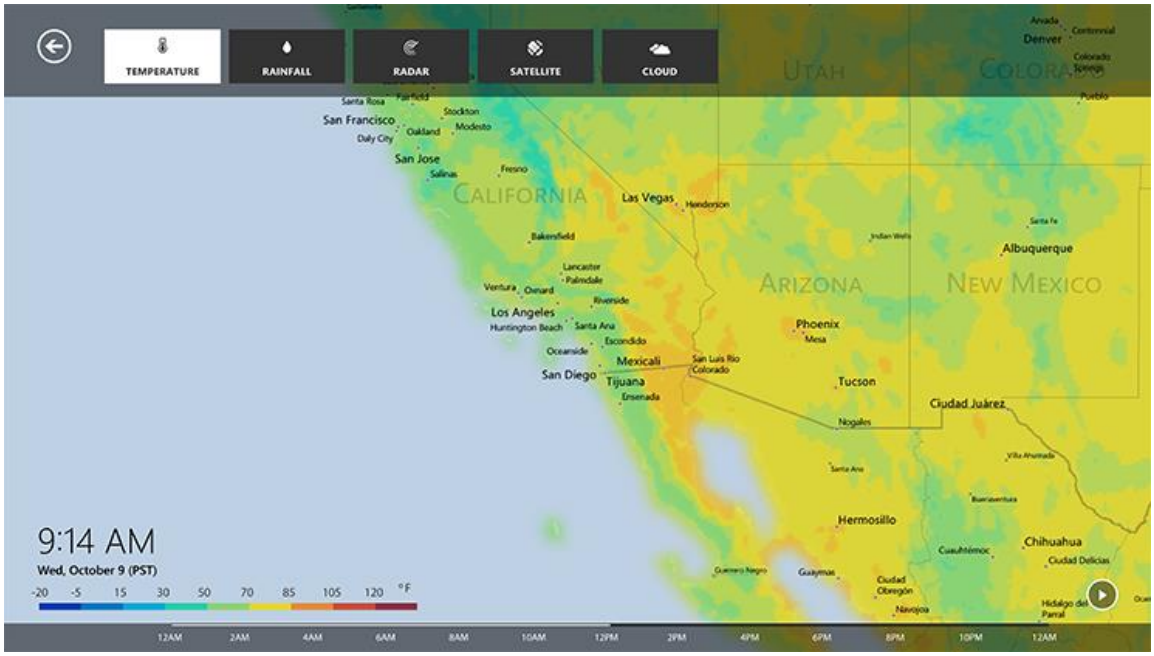
標準のエクスペリエンスから離れてアプリ バーをカスタマイズする場合は、XAML の **CommandBar** コントロールではなく、**AppBar** コントロールを使います。

コマンド用に組み込みのアイコンを使うことも、独自のアイコンを作成することもできます。利用できるアイコンの一覧については、以下をご覧ください。

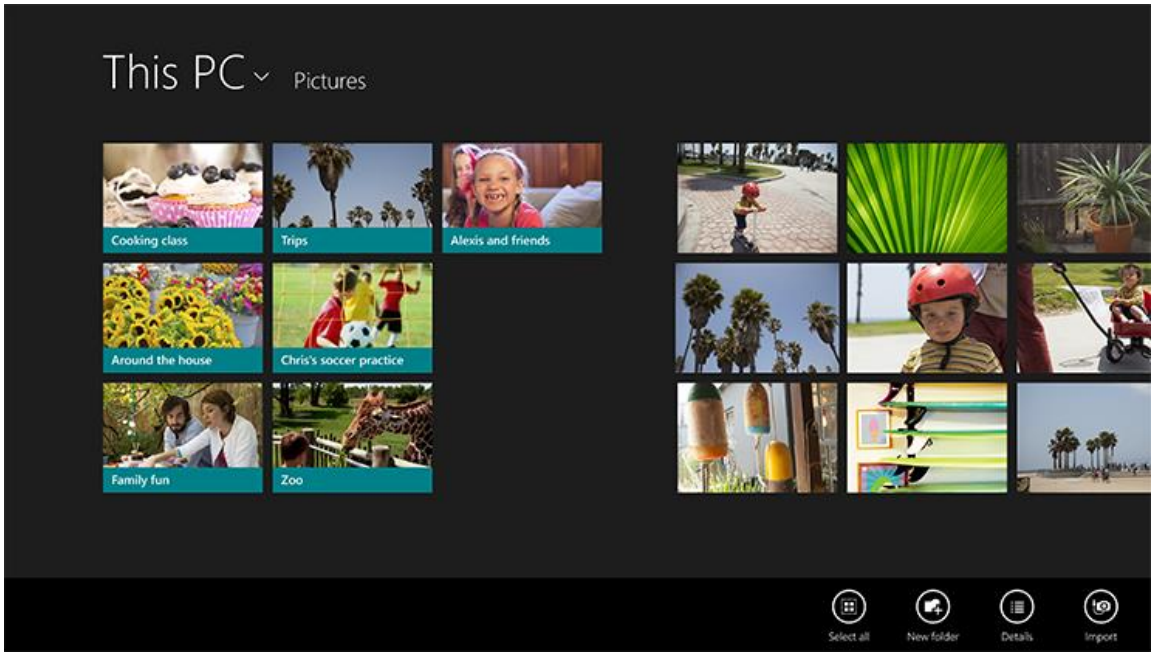
- **AppBarIcon enumeration (Windows Store apps using JavaScript and HTML)**
- **Symbol enumeration (Windows Store apps using C#/VB/C++ and XAML)**

例

上部のアプリ バー



下部のアプリ バー

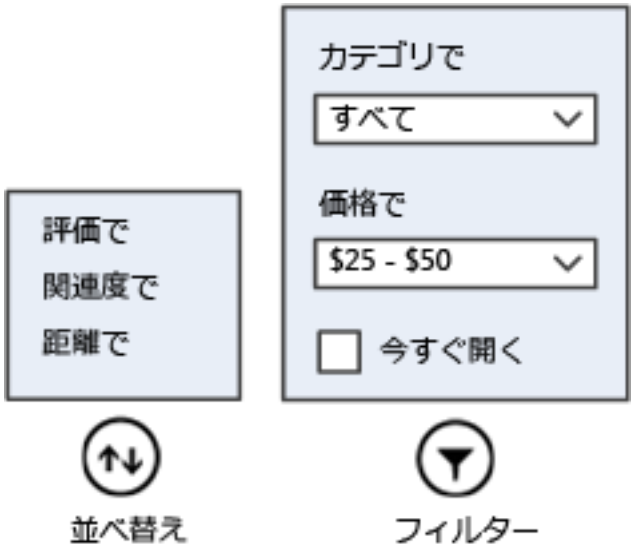


推奨と非推奨

- ナビゲーションとコマンドを分けます。コマンドは下部のアプリ バーに、ナビゲーションは上部のアプリ バーに配置するのが理想です。アプリの初回実行時にヘルプを表示して、アプリ バーに含まれる重要なコマンドをユーザーに知らせることを検討してください。

- 明確に区別できるコマンドのセット (新しいコンテンツを作るためのセットと、ビューをフィルター処理するためのセットなど) がある場合は、アプリ バーの右側と左側に 1 セットずつ配置します。コマンド セットが 3 つ以上ある場合は、区切り記号を使ってセットを区切ります。
- アプリ全体でコマンドをいつも同じ場所に表示します。各ページにはそのページに関連したコマンドのみを含める必要がありますが、ページ間で共通するコマンドは、ユーザーがコマンドの位置を予測できるように、できるだけ同じ場所の近くにコマンドを表示してください。
- 配置規則に従います。
  - [New] (新規)、[Add] (追加)、[Create] (作成) ボタン (とアイコン) は右端に配置します。
  - 表示切り替えのボタンはグループ化して左端に配置します。
  - [Accept] (承諾)、[Yes] (はい)、[OK] (OK) コマンドは、[Reject] (拒否)、[No] (いいえ)、[Cancel] (キャンセル) コマンドの左に配置します。
- 項目を選択したときに状況に応じてアプリ バーにコマンドを表示し、そのアプリ バーを自動的に表示します。ほとんどのユーザーは右利きのため、ユーザーがアプリ ページで項目を選択すると、その項目に関連するコマンドはすべてアプリ バーの左側に表示されます。これにより、ユーザーの腕や手でコマンドの表示が隠れることはありません。
- アプリ バーに状況依存のコマンドがある場合は、ユーザーがアプリを操作しているときにバーが自動的に非表示にならないように、その状況が続いている間はモードを固定に設定します。その状況が終わったら、固定モードを解除します。
  たとえば、ユーザーが画像を選んだときに写真操作に関する状況依存のコマンドを表示しますが、画像の回転やトリミングなどの画像操作をユーザーが続けられるようにするとします。この場合、ユーザーが画像の選択を解除するかエッジ (端) をスワイプしてアプリ バーを閉じるまで、バーは表示されたままです。

- すべてのコマンドを個別のボタンとしてアプリ バーに配置しきれない場合は、類似性のあるコマンドをグループ化して、ユーザーがアプリ バーのボタンからアクセスできるメニューに配置します。コマンドは論理的にグループ化します。たとえば、[返信]、[全員に返信]、[転送] を [応答] メニューに配置します。



- サイズ変更と縦向きビューに対応したアプリ バーを設計します。コマンドが 10 個以下の場合は、ユーザーがアプリをサイズ変更したり縦向きにしたりすると自動的にラベルが非表示になり、余白が調整されるため、コマンドのツールチップを示します。カスタム ビューがもう 1 つ必要な場合は、コマンドをグループ化してメニューにするか、より対象を限定したエクスペリエンスを用意して、スナップ ビューまたは縦向きのビューのコマンドを少なくすることができます。

  JavaScript アプリでは、ユーザーがアプリをスナップしたときに最適なレイアウトになるように、DOM 内の状況依存のコマンドの前にグローバル コマンドを配置することをお勧めします。

  C#/C++/VB アプリの場合、**CommandBar** コントロールを使うと、サイズ変更は自動的に処理されます。
- アプリ ページの下部に水平スクロール領域があるアプリでは、アプリ バーが固定モードで表示されている場合にスクロール領域の高さを縮小します。そうしないと、アプリ バーでスクロール バーが隠れるため、ユーザーは、引き続きスクロールできるようにアプリ バーを消さなければならない場合があります。アプリ バーの上端に対して、スクロール バー フラッシュの下端の位置を維持するよう試みてください。

- 重要なコマンドはアプリ バーには配置しません。たとえば、カメラ アプリでは、アプリ バーではなくアプリ ページに "画像の撮影" コマンドを配置します。写真を撮るためにアプリ ページにボタンを追加するか、ユーザーにプレビューをタップさせることができます。
    - ログイン、ログアウトなどのアカウント管理コマンドは、アプリ バーには配置しません。ログイン、ログアウト、アカウント設定、アカウント作成などのすべてのアカウント管理コマンドは、設定フライアウトに配置する必要があります。ユーザーが特定のページにログインすることが重要な場合は、アプリ ページにログイン ボタンを用意します。詳しくは、「ログインのガイドライン」をご覧ください。
- アプリの設定はアプリ バーに配置しません。既定値や基本設定などのすべてのアプリ設定コマンドは、設定フライアウトに配置する必要があります。また、設定フライアウトは、履歴を消去するためのコマンドなど、あまり使われない管理コマンドを配置するのに最適な場所です。

その他の使い方のガイダンス

さまざまなウィンドウ サイズに合わせたスケーリング

ユーザーがアプリ バーのあるウィンドウのサイズを変更すると、コマンドのサイズが変更され、ラベルが省略されることがあります。縮小サイズのコマンドは画面の 1 つの行に収まらないため、2 行目に折り返されます。

- アプリ バーについては、少なくとも 2 種類のビューを設計してください。フル サイズのビューと、縮小サイズのビューの 2 つです (最小で 500px か 320px)。ほとんどのユーザーは、一般的なウィンドウ サイズ (全画面か、別のアプリと半々) のみを使います。
- 小さいビューを設計するときはコマンドをグループ化します。コマンドを意味のあるグループに分けられない場合は、省略記号のアイコンを使って "その他" というグループに配置します。

一般的なウィンドウの解像度 (ピクセル単位)	単一行に表示できる、ラベルのない縮小サイズのボタンの数	単一行に表示できる、ラベルのあるフル サイズのボタンの数
1366	22	13

1024	16	10
768	12	7
500	8	5
320	5	3

マウスの右ボタンの処理

アプリの UI を他の Windows ストア アプリと同じようにするには、ユーザーがマウスの右ボタンをクリックしたときに、用意したアプリ バーがトリガーされる必要があります。しかし、アプリでマウスの右ボタンを別の目的 (ゲームでのサブ武器、3-D ビューアーでの仮想トラックボールなど) で使う場合、アプリはアプリ バーを起動するイベントを無視できます。それでも、アプリ バーは Windows ストア アプリのエクスペリエンスの重要な部分なので、ゲームのコントロール モデル内でのアプリ バーや同様のコンテキスト メニューの役割について検討する必要があります。

アプリのコントロールを設計するときは、次のガイドラインに従ってください。

- アプリの重要な機能のためにマウスの右ボタンを使う必要がある場合は、右ボタンでその機能を直接呼び出します。コンテキストに沿った UI やアプリ バーは、ワークフローにとって重要でなければ、アクティブ化しません。
- 境界線メニューのような、コンテキストに沿った右クリック アクションを必要としない領域を右クリックしたら、アプリ バーを表示します。
- これらのソリューションが不十分な場合は、カスタム ユーザー操作コントロールを使って、マウス操作でアプリ バーを開けるようにします。
- **MouseMoved** などの **MouseDevice** クラス イベントを使って、独自のコンテキスト メニューの動作を実装します。
    - タッチ操作での長押しは、右ボタンのクリックに相当することを忘れないでください。両方のイベントを同じように処理します。このイベントを処理し、カスタム動作を定義するには、**Holding** イベントに登録します。長押しを有効にするには、**Hold** (タッチ入力とペン入力の場合) と **HoldWithMouse** を **GestureSettings** プロパティで設定します。
- Windows キーを押しながら Z キーを押す動作を変更しないでください。アプリでは、アプリ バーまたはコンテキスト メニューを表示します。これらの 2 つのキー

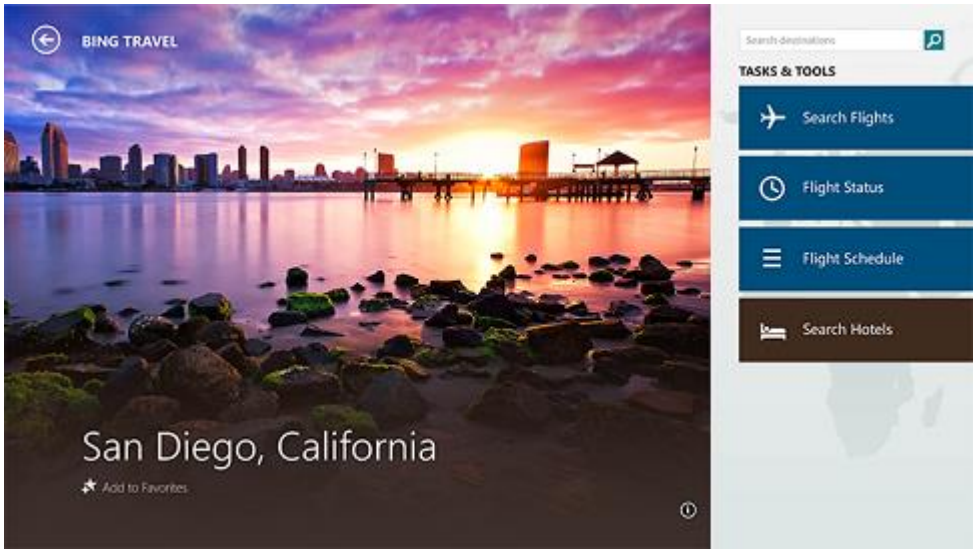
が押されたかどうかを判断するために、**KeyDown** と **AcceleratorKeyActivated** イベントに登録します。

戻るボタンのガイドライン

説明

戻るボタンは、戻るナビゲーションをボタンの形式で提供します。

例



推奨と非推奨

- 階層型ナビゲーション パターンを使うアプリでは、戻るボタンを使います。
- アプリの幅が狭い場合、小さい戻るボタンを使います。
- フラット ナビゲーション パターンを使うアプリでは、戻るボタンを使わないでください。フラット ナビゲーション アプリでは、ユーザーは通常コンテンツ内の直接リンクかナビゲーション バーを使ってページ間を移動します。

ボタンのガイドライン

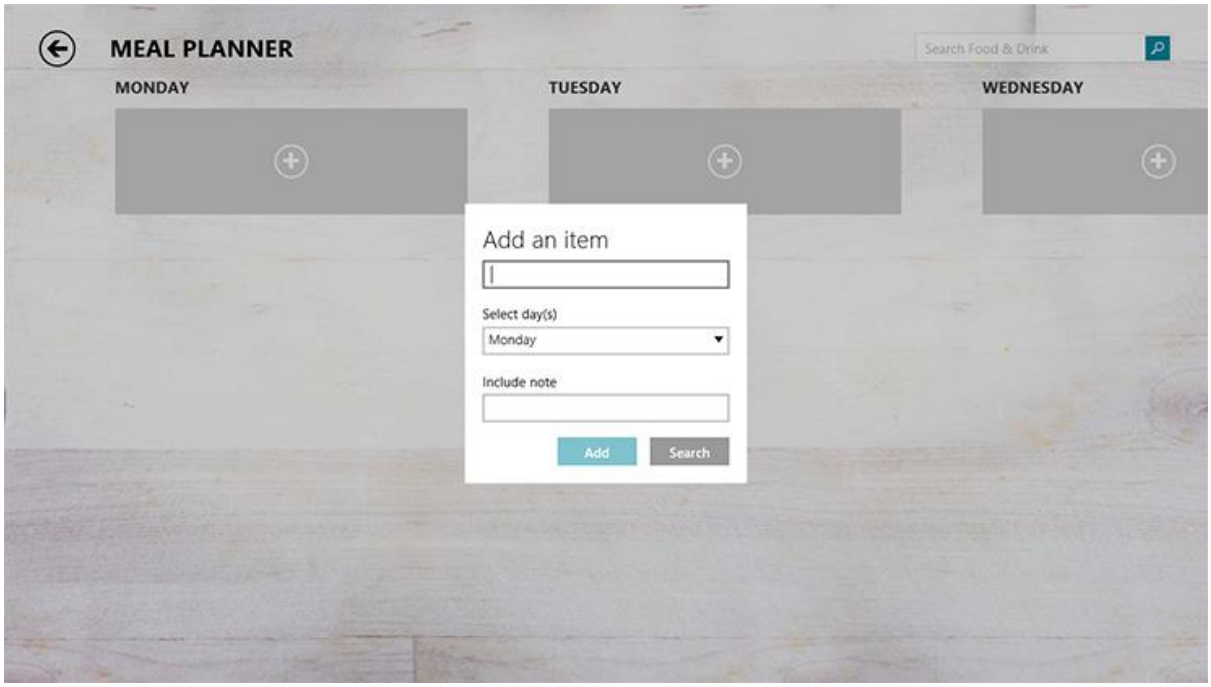
Windows アプリ：ボタンの状態

Windows Phone アプリ：ボタンの状態

説明

ボタン (コマンド ボタン) は、特定の操作を直ちに実行する方法をユーザーに与えます。

例



適切なコントロールの選択

ボタンを使うと、ユーザーは直ちに操作を開始できます (フォームの送信など)。

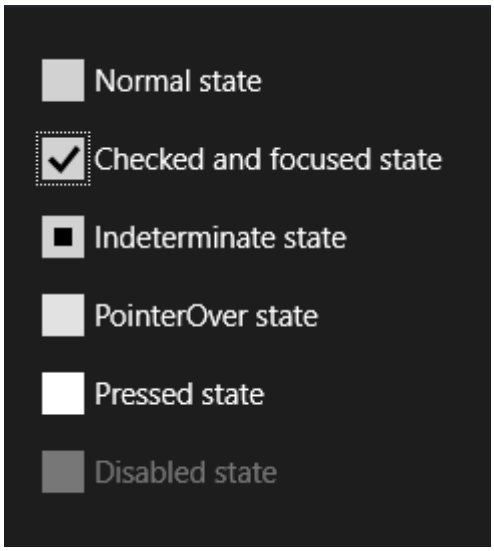
他のページに移動する操作では、ボタンは使わず、リンクを使います。 例外: ウィザードでのページの移動には、[戻る] と [次へ] というラベルのボタンを使います。他の種類の前に戻る移動や上位レベルへの移動では、win-backbutton スタイルのボタンを使います。

推奨と非推奨

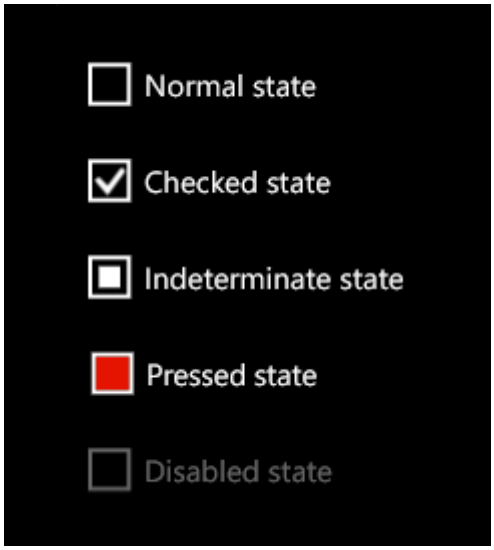
- ボタンの用途と状態をユーザーがはっきりと理解できるようにします。
- ボタンによって行われる操作を明確に説明する、簡潔で具体的でわかりやすいテキストを使います。通常、ボタンのテキスト コンテンツは、1 語の動詞です。
- ボタンのテキスト コンテンツが動的な場合 (ローカライズされる場合など) は、ボタンのサイズがどのように変化し、その周囲のコントロールに何が起こるかを考えます。
- テキスト コンテンツの付いたコマンド ボタンの場合は、最小のボタン幅を使います。
- テキスト コンテンツの付いた幅が狭い横長または縦長のコマンド ボタンは使わないようにします。

- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- ある操作をアプリの複数のページで実行できるようにするには、各ページでボタンを使うのではなく、下部のアプリ バーを使うことを検討します。
- フォームの送信に AJAX を使う場合は、送信ボタンを使ってフォームの送信機能を上書きして、フォームのどこにフォーカスがあっても、ユーザーが Enter キーを押すとコミットできるようにします。
- ユーザーに対して表示するボタンは、1 つまたは 2 つにします (例: [承諾] と [キャンセル])。3 つ以上の操作をユーザーに示す必要がある場合は、操作を開始する 1 つのコマンド ボタンをユーザーが選択できるチェックボックスまたはラジオ ボタンを使うことを検討します。
- 最も一般的な操作や推奨される操作を示す既定のコマンド ボタンを使います。
- ボタンをカスタマイズすることを検討します。ボタンの形は既定では四角形ですが、ボタンの外観を構成するビジュアル効果をカスタマイズできます。ボタンのコンテンツは、通常はテキスト (―例: [承諾] や [キャンセル]―) ですが、アイコンに置き換えるか、アイコンとテキストを使うことができます。
- ユーザーがボタンを操作したとき、ボタンの状態と外観を変更して、ユーザーにフィードバックを返します。ボタンの状態には、Normal、Pressed、Disabled などがあります。
- ユーザーがボタンをタップまたはクリックしたときに、ボタンのアクションを開始します。通常、アクションは、ユーザーがボタンを離したときに開始されますが、指がボタンを押したときにボタンのアクションを開始するように設定することもできます。
- コマンド ボタンは、状態の設定には使わないでください。
- ボタンのテキストは、ローカライズ以外の目的では変更しないでください。
- 送信、リセット、標準ボタンの既定のスタイルを取り替えないでください。
- ボタン内に多すぎるコンテンツを配置しないでください。表やチェック ボックスなど、他の大半の HTML 要素を含めることができますが、多すぎるコンテンツはユーザーを混乱させることになります。ボタン内のコンテンツは、簡潔でわかりやすくします (画像と少しのテキストのみにします)。

チェック ボックス コントロールのガイドライン



Windows アプリ：チェック ボックスの状態



Windows Phone アプリ：チェック ボックスの状態

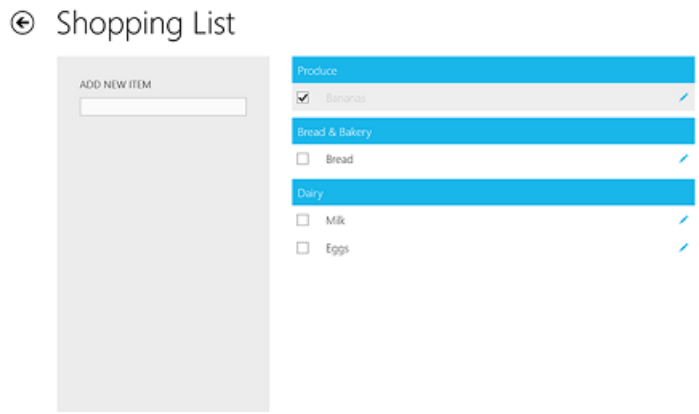
説明

チェック ボックスでは通常、ユーザーが 2 つの独立した正反対のオプションから 1 つを選択できます。たとえば、ニュースレターを購読するかどうか、一覧にあるメールを個別に選択するかどうかなどを選択します。まれに、チェック ボックスがオン状態でもオフ状態でもない "不確定状態" という状態になることがあります。たとえば、チェック ボックスが、混在する値を持つサブ選択のコレクションを表すときに、チェック ボックスのオン状態が不確定状態になります。

ユーザーがチェック ボックスを操作すると、チェック ボックスは状態 (外観) を変更してフィードバックを返します。チェック ボックスの状態の例としては、標準、押された状態、オフの状態、オンの状態などがあります。

ユーザーはチェック ボックスをタップして反対の値に切り替えます。

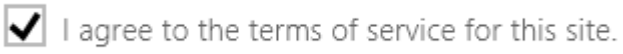
例



適切なコントロールの選択

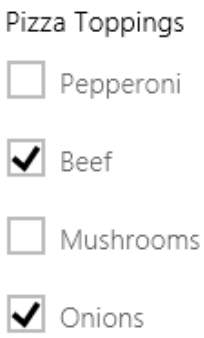
ユーザーに二者択一、相互排他的でない 1 つ以上のオプション、混在する選択を提示するには、チェック ボックスを使います。

"はい" か "いいえ" の選択には、1 つのチェック ボックスを使います。



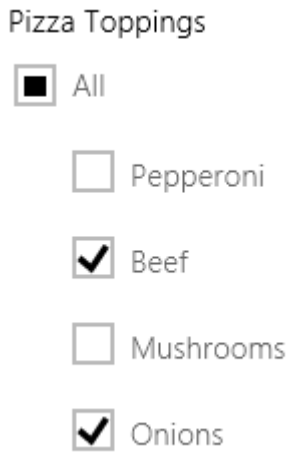
二者択一の場合、チェック ボックスとトグル スイッチとの主な違いは、チェック ボックスが状態を管理し、トグル スイッチが動作を管理する点です。 チェック ボックスによる操作はコミットを遅らせることができますが (たとえばフォームの送信の一部として)、トグル スイッチによる操作は直ちにコミットしなければなりません。また、複数の選択ができるのは、チェック ボックスだけです。

ユーザーがオプションの任意の組み合わせを選べる場合は、チェック ボックスのグループを作成します。



ラジオ ボタンのグループが 1 つの選択を表すラジオ ボタンとは異なり、グループ内の各チェック ボックスが個別の独立した選択を表します。1 つ以上のオプションがあっても、選択できるのが 1 つだけの場合は、代わりにラジオ ボタンを使います。

オプションが複数のオブジェクトに適用される場合は、チェック ボックスを使って、オプションの適用対象がすべてのオブジェクトか、一部のオブジェクトか、いずれにも適用されないかを示すことができます。オプションの適用対象がすべてのオブジェクトではなく、一部のオブジェクトである場合は、混在する選択を表すために、チェック ボックスの中間的な状態を使います。 混在する選択のチェック ボックスの 1 つの例は、すべてでなく一部のサブ項目をユーザーが選んだ場合に中間的な状態になる [すべて選択] チェック ボックスです。



詳しくは、次のトピックをご覧ください。

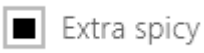
- **indeterminate property (HTML)**
- **Indeterminate event (XAML)**

推奨と非推奨

- チェック ボックスの用途と現在の状態が明確であることを確認します。チェック ボックスをオフにする意味が明確であることを確認します。たとえば、横向きに対してチェック ボックスを使う場合、チェック ボックスをオフにすることが縦向きを意味するかどうかは明確ではありません。その場合は、代わりに [横] と [縦] の 2 つのラジオ ボタンを使います。
- チェック ボックスのテキスト コンテンツは 2 行以内にします。
- チェック マークをオンにすると true、チェック マークをオフにすると false に設定されるようにテキスト コンテンツを記述します。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- 提示する選択が複数ある場合は、レイアウト パネルが組み込まれたスクロール ビューアー コントロールを使うことを検討してください。
- チェック ボックスのラベル内にチェック ボックスを配置して、ラベルをクリックするとチェック ボックスの状態が切り替わるようにします。こうすることで、選択領域のサイズが広がり、タッチ ユーザーがチェック ボックスを使いやすくなります。
- すべてではなく一部の子オブジェクトのためにオプションが設定されていることを示すために、不確定の状態を使います。
    - 不確定の状態を使う場合は、下位のチェック ボックスを使って、どのオプションが選択され、どのオプションが選択されていないかがわかるようにします。ユーザーにサブ選択肢がわかりやすいように UI を設計します。
- テキスト コンテンツが動的な場合、コントロールのサイズがどのように変わり、周囲のビジュアル効果にどのような影響が生じるかを検討してください。
- 2 つのチェック ボックスのグループを並べて配置しないようにします。並べて配置すると、どのオプションがどのグループに属しているのかが、ユーザーにわかりにくくなります。グループを分けるには、グループ ラベルを使います。

- オン/オフの制御やコマンドの実行にはチェック ボックスを使わず、代わりにトグル スイッチを使います。
- ダイアログ ボックスなどの他のコントロールを表示するためにチェック ボックスを使わないでください。
- 不確定の状態を、第 3 の状態を示すために使わないでください。不確定の状態は、すべてでなく一部の子オブジェクトのためにオプションが設定されていることを示す目的で使います。そのため、ユーザーが不確定の状態を直接設定できないようにします。

  良くない例を示すために、次のチェック ボックスでは不確定の状態を使って "中辛" を示しています。

  このような場合は、[Not spicy]、[Spicy]、[Extra spicy] という 3 つのオプションがあるラジオ ボタン グループを使います。

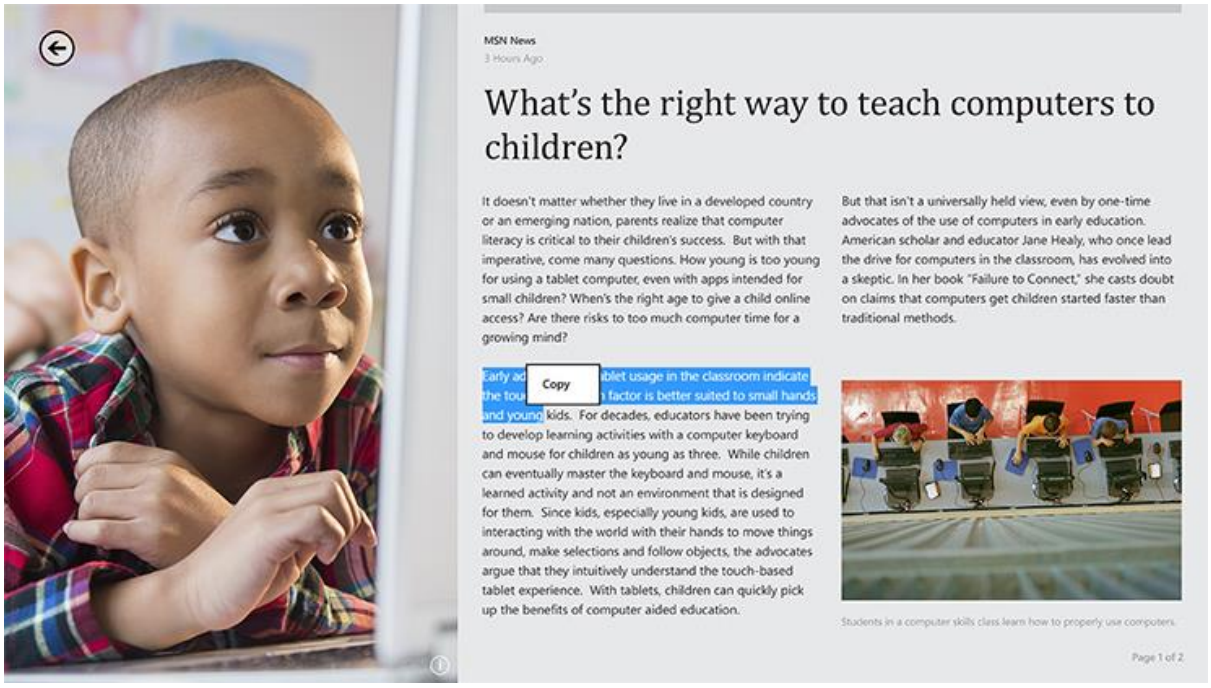
  

コンテキスト メニューのガイドライン

説明

コンテキスト メニューは、ユーザーがテキストに対する操作 (クリップボード コマンドなど) をすばやく実行できるようにするための簡易メニューです。アプリでは、システムに用意されているテキストとハイパーリンクに関する既定のコンテキスト メニューを使うことができます。 既定のコンテキスト メニューを、テキストまたはハイパーリンクのカスタム コマンドを表示するメニューに置き換えることができます。

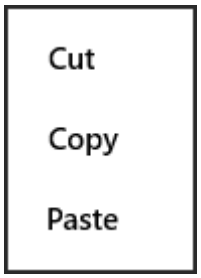
例



推奨と非推奨

- カスタム コマンドの表示します。システムには、テキストとハイパーリンクに関する既定のコンテキスト メニューが用意されています。アプリでは、これらのコンテキスト メニューを独自のコンテキスト メニューに置き換えることができます。カスタム コマンドをコンテキスト メニューに含めるのが適切なのは、アプリのツール バーには存在せず、直接的な操作 (回転など) を使って行うことができない場合です。

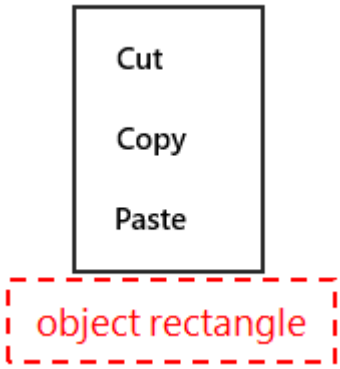
- コンテキスト メニューを使って、選択されたテキストなどのオブジェクトのクリップボード コマンド (切り取り、コピー、貼り付け) を表示します。既定では、選択されたテキストの [切り取り]、[コピー]、[貼り付け] コマンドが、システムによって表示されます。一般的な貼り付けメニュー コマンドは、[すべて選択]、[貼り付け]、[元に戻す] です。これらのコマンドを上書きするには、カスタマイズされたコンテキスト メニューを表示します。できれば、既定のコマンドは保持して、アプリ内でもシステムと同様の動作が行われるようにします。



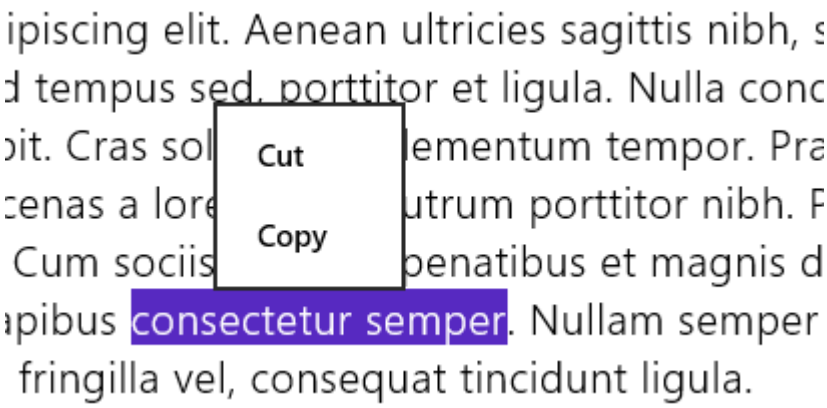
- 操作する必要があるものの、選択できなかったり、他の方法で指定できなかったりするオブジェクトのためのコマンドを表示するには、コンテキスト メニューを使います。例: チャットの会話は、選択対象の追加先として適していません。この場合、コンテキスト メニューであれば、チャットの会話の各メッセージでコマンドを使用可能にできます。

- コマンド名は短くします。コンテキスト メニューでは、最大幅が約 50 文字です。コマンド名が長すぎる場合は、自動的に切り詰められ、表示されなくなった文字の代わりに省略記号 ("…") が表示されます。コンテキスト メニューでコマンド名が切り詰められるのを避けるために、コマンド名は短くします。
- 各コマンド名には文としての大文字表記を使います。コマンドの最初の文字を大文字にし、残りのすべての文字は小文字にします。
- 関連するコマンドのグループを区別するために、区切り線を使います。コンテキスト メニュー内で区切り線を使い、コマンドのセットをグループ化します。一緒に表示される、アプリ固有またはビュー固有のコマンドと、一般的なコマンド (クリップボード コマンドなど) のセットを区別するには、区切り線を使います。
- 表示するコマンドをできるだけ少なくして、最大でも 6 個までにします。
  メニューにコマンドを収めるのが難しい場合は、次の点を確認します。
  - このコマンドは直接的な操作で行うことができるか?
  - このコマンドがなかった場合、ユーザーのエクスペリエンスはどのようになるか?
  - このコマンドは、他の方法で使用できるか?　コマンドをコンテキスト メニューにも重複させることにどのような利点があるか?
  - 項目には、それを表すページがあるか?　ある場合は、コレクションのコンテキスト メニューではなく、そのページのアプリ バーまたはキャンバスにコマンドを含めることができます。
  - このコマンドは、常に表示する必要があるか、それとも特定の状況だけで必要か?
- コンテキスト メニューのカスタム コマンドは、重要度の高いコマンドが下になるように並べます。

- クリップボード コマンドは、標準の [切り取り]、[コピー]、[貼り付け] の順で、メニューの下端に並べます。2 つのクリップボード コマンドだけ (たとえば [切り取り] と [貼り付け]) を表示する場合は、使わないコマンドを単に除外し、それ以外の順序は維持します。
- ユーザーの操作対象のオブジェクトの近くにコンテキスト メニューを表示するコンテキスト メニューは、操作対象のオブジェクトまたは選択項目の近くに表示します。コンテキスト メニューを表示するときには、操作対象のオブジェクトを示す四角形を用意し、その近くにコンテキスト メニューを表示します。既定では、コンテキスト メニューは操作対象のオブジェクトの上方中央に表示されます。



- コンテキスト メニューを消します。コンテキスト メニューを表示する状況でなくなったら、プログラムを使ってコンテキスト メニューを消します。そのためには、標準の非同期パターンの取り消しを使います。
- 項目のアクセス キーを使わないでください。コンテキスト メニュー コマンドでは、アクセス キーを使えません。メニューで、コマンド名の先頭にアンパサンド (&) を使いません。
- 選択対象やオブジェクトと状況的な関連性がない場合は、コンテキスト メニューにコマンドを表示しないでください。コンテキスト メニューには、無効化の状態がありません。コマンドが含まれていない場合に、表示されなくなります。
  たとえば、ユーザーが編集できるテキストに貼り付けられるテキストがクリップボードにない場合、既定のコンテキスト メニューには **[貼り付け]** コマンドが表示されません。代わりに、**[切り取り]** と **[コピー]** コマンドだけが表示されます。

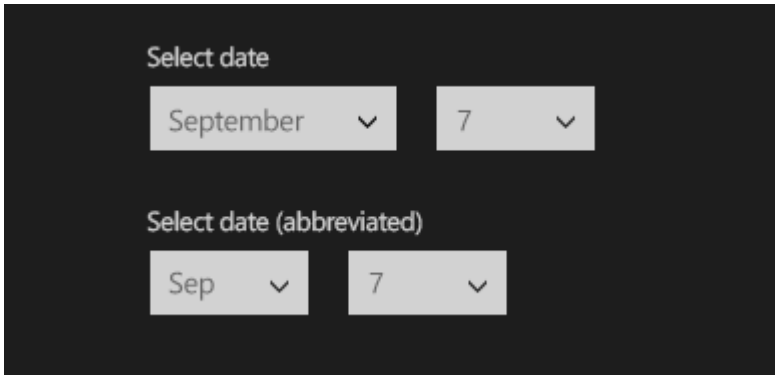


- 直接的な操作または選択が可能な場合は、コンテキスト メニューにコマンドを追加しないでください。ユーザーは、基本的に、UI 要素を直接操作するか、UI 要素を選択してアプリ バー上のコマンドを操作することで、コマンドを実行するべきです。そうすることで、画面上でのコマンドの場所がわかりやすく、見つけやすくなります。たとえば、ユーザーが写真を回転するときには、コンテキスト メニューの ["回転"] コマンドを使うのではなく、画像を指で直接操作できるようにする必要があります。
- コマンドをコンテキスト メニューに重複しなようにしてください。直接的な操作のような明らかな操作方法や、既存のアプリ バー コマンドが既にある場合は、同じ操作のためのコンテキスト メニュー コマンドを追加しません。重複させるよりも、ユーザーが項目を選択したり操作対象の項目に移動したりして、コマンドを見つけられるようにします。ただし、キーボード ショートカットで実行できる一部の操作は例外です。操作がキーボード ショートカットだけで実行できる場合 (Ctrl + C によるコピーなど) は、コマンドをコンテキスト メニューに追加してその操作が重複しても問題ありません。
- ページの背景や大きなオブジェクトのコンテキスト メニューを表示しないでください。ページの背景や、画面全体を占めるようなオブジェクトを操作するコマンドがある場合は、代わりにアプリ バーを使うか、コマンドをアプリのキャンバスに追加して、ページまたはオブジェクトを操作します。
- エラーになるコマンドは表示しないでください。たとえば、貼り付ける場所が編集できない場合は、貼り付けコマンドを表示しません。

その他の使い方のガイダンス

コンテキスト メニューは、通常は表示されないため、画面領域を節約するうえで効率的な手段です。選ばれたオブジェクトやウィンドウの領域に適用される状況依存コマンドとオプションがいくつかある場合は、コンテキスト メニューを使います。コンテキスト メニューには特定の順序があります。最もよく使われる項目 (プライマリ コマンド) が先頭で、転送コマンドがその後に続き、プロパティは最後です。この順序が、効率性と予測可能性を高めます。

DatePicker のガイドライン

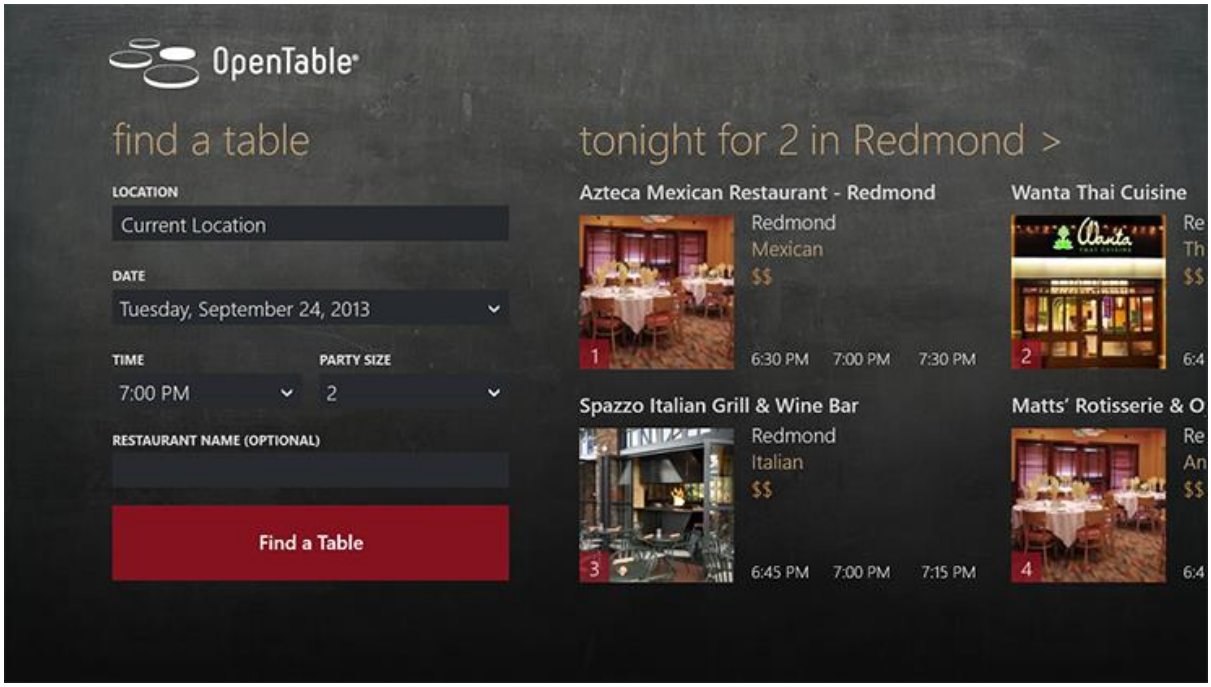


説明

DatePicker は、ユーザーがタッチ、マウス、またはキーボード入力を使ってローカライズされた日付を選択できる標準化された方法です。

ユーザーが単一の日付を選ぶ必要がある場合は、日付セレクターを使います。DatePicker は、使われる画面領域が固定されていて選択の数とは関係がないため、領域を節約するうえで優れたオプションです。

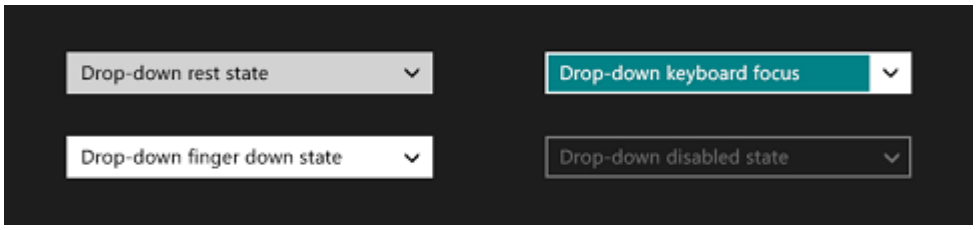
例



推奨と非推奨

- DatePicker は、フォームに日付を表示したり、領域を効率的に使う必要があったりするときに使います。
- 200 年を超えるような年の範囲は設定しません。 代わりに、最小年と最大年のプロパティを使って、有効な年の範囲を定義します。
- 日付範囲を選ぶ場合は、DatePicker を使わないでください。
- コマンドを実行する場合、ダイアログ ボックスなどの別のウィンドウを表示する場合、または別のコントロールを動的に表示する場合には、DatePicker を使わないでください。

ドロップダウン リストのガイドライン



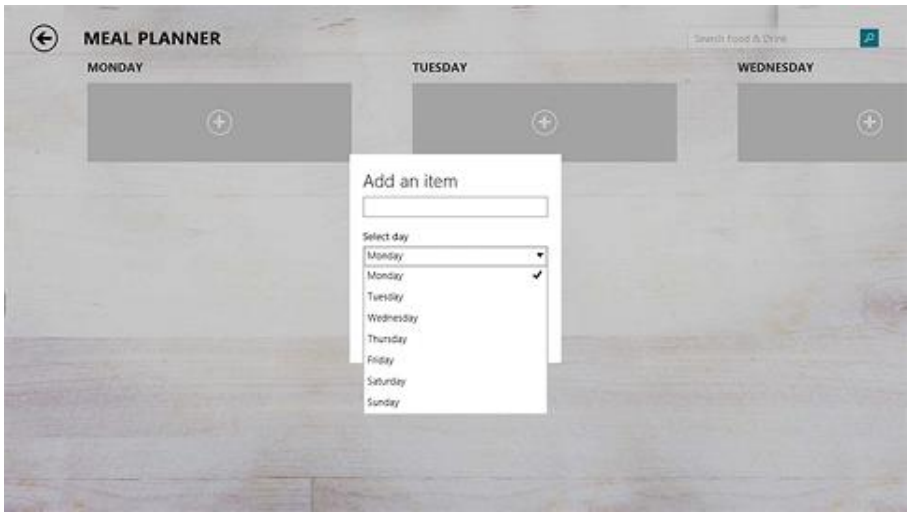
説明

ドロップダウン リストを使うと、ユーザーは値の一覧からオプションを選ぶことができます。ドロップダウン リストは、選ばれた項目が常に表示される一覧です。その他の項目は、ドロップダウン ボタンをクリックすると、オンデマンドで表示されます。HTML では、ドロップダウン リストは、ポップアップ モードの選択コントロールと呼ばれます。XAML では、ドロップダウン リストはコンボ ボックスと呼ばれます。ドロップダウン リストでは、ユーザーは 1 つのオプションのみを選ぶことができます。

また、追加操作なく、すべての項目を表示する一覧を作成できます。単一選択と複数選択の両方がサポートされています。HTML では、この種類の一覧はインライン モードの選択コントロールと呼ばれます。XAML では、この一覧はリスト ボックスと呼ばれます。詳しくは、「リスト ボックスのガイドライン (または選択)」をご覧ください。

例

次に、ドロップダウン リストの例を示します。この一覧の項目は、ユーザーがドロップダウン ボタンを選択したときにのみ表示されます。



これは、一覧のすべての項目が常に表示されるリスト ボックスの例を示しています。

適切なコントロールの選択

1 行のテキストで十分に表すことができる項目のセットから 1 つ以上の値をユーザーが選択できるようにするには、ドロップダウン リスト コントロールを使います。

複数行のテキストや画像を含む項目を表示するためには、このコントロールは使いません。代わりに、リスト ビューまたはグリッド ビュー コントロールを使います。

項目が 5 個より少ない場合は、代わりにラジオ ボタン (1 つの項目だけを選べる場合) またはチェック ボックス (複数の項目を選べる場合) を使うことを検討します。

選択項目がアプリのフローにおいて二次的な重要性しか持たない場合に、ドロップダウン リストを使います。 ほとんどのユーザーのほとんどの状況で既定のオプションがお勧めされている場合は、リスト ボックスを使ってすべての項目を表示すると、オプションに必要以上の注意を引いてしまう可能性があります。ドロップダウン リストを使うことで、領域を節約し、無駄な情報を最小限にすることができます。

推奨と非推奨

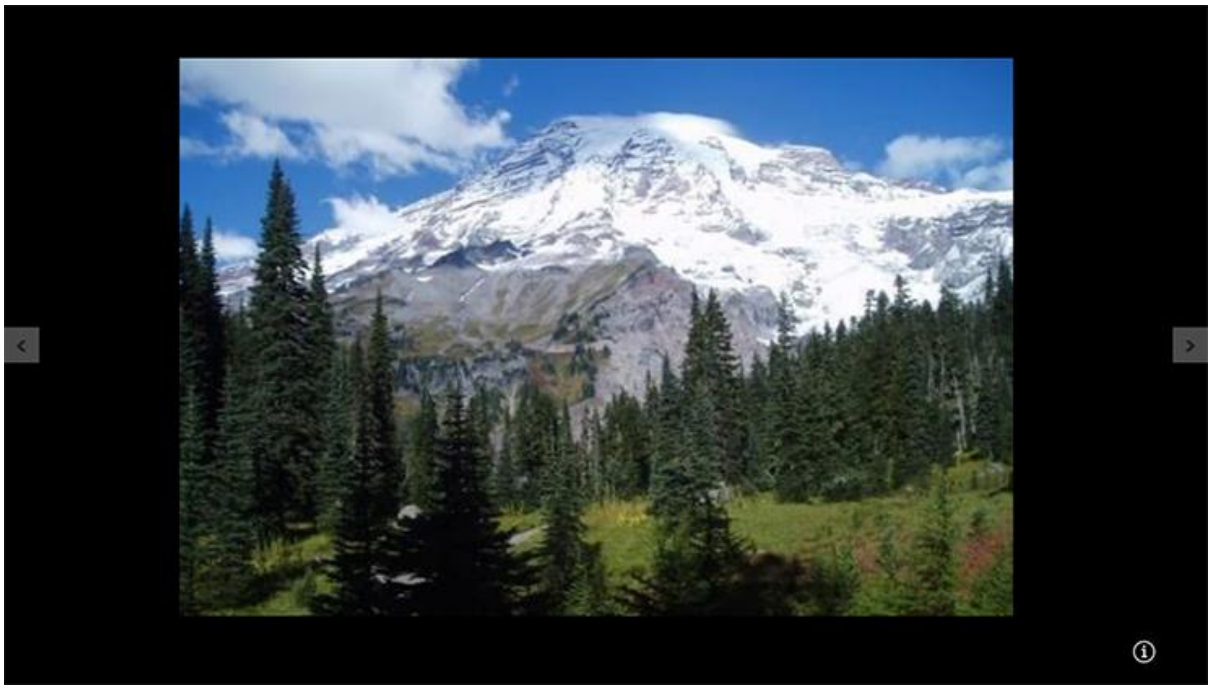
- ドロップダウン リスト項目のテキストのコンテンツは、単一行に制限します。
- 関連するオプションをグループ化する、最も一般的なオプションを最初に配置する、アルファベット順を使うなど、論理的な順序でドロップダウン リストの項目を並べ替えます。名前はアルファベット順、数値は数値順、日付は時系列順に並べ替えます。
- スクロール バーがコンテンツと重ならないように、コンテンツの右側に 27 ピクセルのパディングを追加します。

FlipView コントロールのガイドライン

説明

FlipView コントロールを使うと、アプリ内のビューやコレクション内の項目を、フォト アルバム内の写真のように 1 つずつめくっていくことができます。 ユーザーは、コレクション内を移動しながら、個々の項目を見ていくことができます。

例



適切なコントロールの選択

FlipView コントロールを使うと、ユーザーはビューや項目を 1 つずつめくっていくことができます。マウスをホバーするとフリップ ボタンが表示され、それを使って次の項目または前の項目に移動できます。

ユーザーが特定の項目に直接ジャンプできるように、コンテキスト インジケーター コントロールを追加することもできます。

推奨と非推奨

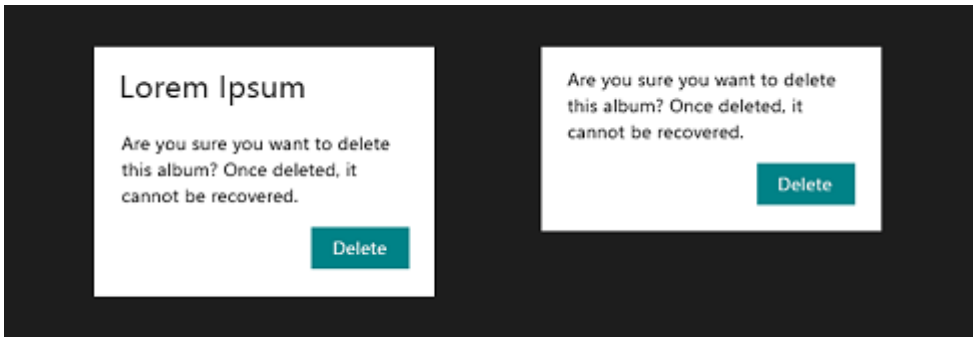
- FlipView コントロールは、次のような場合に使います。
  - 関連する項目の小規模 (< 10) のコレクション内の項目間をフリップする場合。たとえば、ショッピング アプリの商品の詳細ページで、商品のいくつかの画像を表

示したい場合があります。小規模なコレクションでコンテキスト インジケーター コントロールを使うことができますが、通常は必要ありません。

- 関連する項目の中規模 (10 ～ 25) の一覧内の項目間をフリップする場合。たとえば、不動産アプリで、各物件の室内や眺望を示す多数の画像を表示したい場合があります。このような中規模の一覧では、ユーザーが特定の写真にジャンプできるように、サムネイルのフィルムストリップのようなコンテキスト インジケーター コントロールも用意します。
- アプリケーションのビュー間をフリップする場合。たとえば、アプリ起動ツールに、ユーザーのお気に入りをグリッドで表示するビューと、リスト形式で表示するビューと、リスト形式で表示するビューがある場合があります。FlipView コントロールを追加して、ユーザーが 1 つのビューから別のビューに移動できるようにします。

- コレクション内の項目に、コレクション内での位置をユーザーが判断できるだけの十分なコンテキスト情報がない場合に、コンテキスト インジケーターを使います。 曜日の FlipView にコンテキスト インジケーターは不要ですが、製品画像の FlipView には必要でしょう。
- 現在の項目がコレクション内でどの位置にあるかをユーザーに示します。 コンテキスト インジケーターを使って、コレクションがどのくらいの大きさで、現在の項目がコレクションの他の項目との関連の中でどこに位置しているかをユーザーがわかるようにします。
- ユーザーが特定の項目にジャンプできるようにします。 常に、ユーザーが位置と移動を自由に制御できていると感じられるように補助します。
- 大規模なコレクションでは、FlipView コントロールを使わないでください。項目ごとにフリップ操作を繰り返さなければならず、ユーザーの負担になります。代わりに、以下のいずれかを使います。
  - **ListView (JavaScript)**
  - **ListView (C#/C++/VB)**
  - **GridView (C#/C++/VB)**

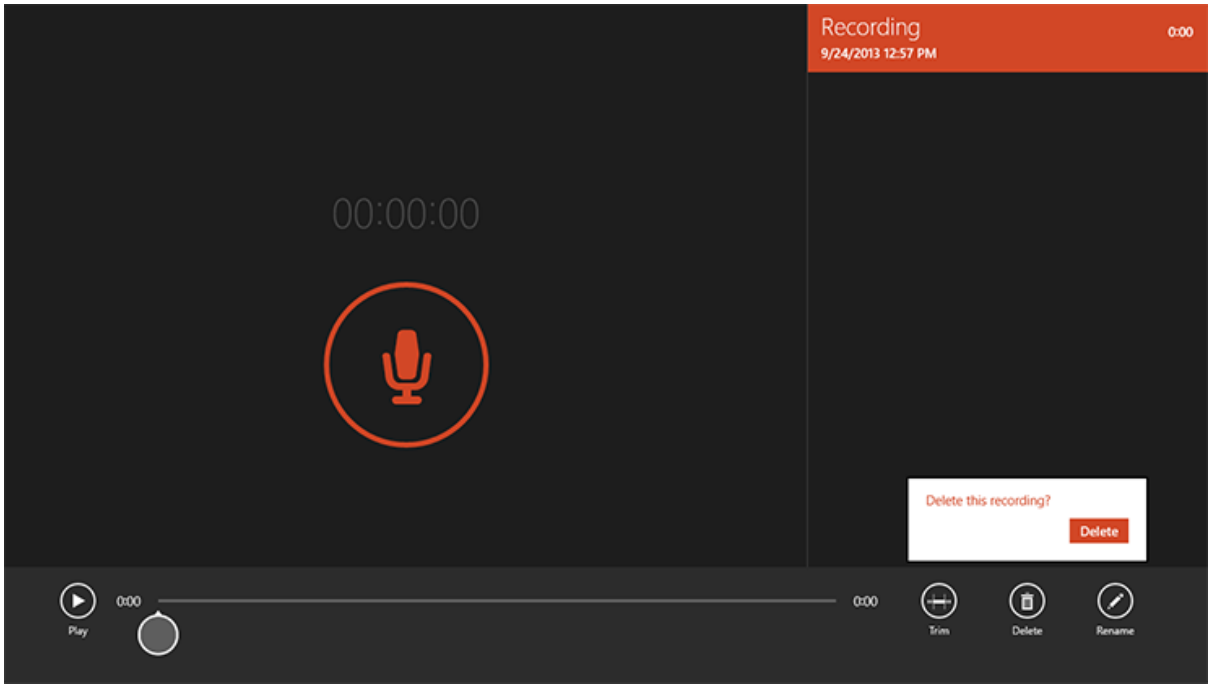
フライアウトのガイドライン



説明

フライアウト (flyout) は、ユーザーが現在操作している内容に関係する UI を一時的に表示するために使われる軽量なポップアップです。このフライアウトは、メニューを表示する場合、非表示コントロールを表示する場合、項目の詳細を表示する場合、またはユーザーに操作の確認を求める場合に使うことができます。フライアウトは、ユーザーのタップまたはクリックのみに応じて表示する必要があります。フライアウトは、ユーザーがフライアウトの外側をタップしたときには必ず閉じます。

例



推奨と非推奨

- フライアウトは、画面上に常に表示しておくのではない UI を表示するのに便利です。ユーザーは、フライアウトの外側をタップまたはクリックするか、Esc キーを押せば、いつでもフライアウトを閉じることができます。ユーザーがフライアウトで選択を行ったら、フライアウトは消えるようにします。
- フライアウトは、呼び出した位置の近くに配置します。ユーザーがツール バー ボタンをタップしてフライアウトを呼び出した場合は、そのツール バー ボタンの上か下にフライアウトを表示します。フライアウトをコントロールの上か下に表示すると重要なコンテンツが隠れてしまう場合は、コントロールの左か右に配置できます。フライアウトを実装する場合は、フライアウトの固定先とするオブジェクトと、そのオブジェクトのどの側面にフライアウトを表示するかを指定して、フライアウトを配置します。フライアウトは、固定先が画面の端にある場合 (スタート画面のユーザー タイル フライアウトなど) を除き、固定先に対して 中央揃えにします。
- フライアウトは、次の目的のために使います。
    - **情報の収集:** ユーザーが選択した操作で、より詳細な指定 (オプションの選択や情報の入力) が必要な場合、その UI をフライアウトに配置し、ユーザーは元のコンテキストにとどまるようにすることができます。 たとえば、地図アプリがあるとします。ユーザーは、タグを付ける場所にラベルを指定できます。タグを付ける場所をユーザーがタップすると、アプリによってフライアウトが表示され、ユーザーはラベルを入力できます。

      例: ブラウザーで、スタート画面に項目をピン留めするには、ユーザーはアプリ バーのピンのアイコンをタップします。次に、フライアウトで新しいタイルの名前を入力します。
    - **警告と確認:** 被害が発生するかもしれない操作をユーザーが行う前に警告します。

      例: 写真アプリで、ユーザーがツール バーの削除アイコンを押したとします。ツール バーのボタンの横にフライアウトが表示され、写真が完全に削除されるという警告と、削除コマンドが表示されます。ユーザーは、表示された削除コマンドを押すことも、削除アイコンを間違って押した場合はフライアウトを消すこともできます。

注 警告またはエラーをフライアウトで表示するのは、それらがすぐに表示でき、ユーザーの操作の直接の結果である場合だけです。詳しくは、「適切な UI サーフェスの選択: エラー」をご覧ください。

- **ドロップダウン メニュー:** アプリ バーのボタンに複数のオプションがある場合は、フライアウトの中にドロップダウン メニューを表示して、ユーザーがオプションを選択できるようにします。

  例: メール アプリで、ユーザーがアプリ バーの [対応] を押したとします。メニューが表示され、ユーザーは対応方法を [返信]、[全員に返信]、[転送] の中から選択できます。ユーザーがアプリ バーの [キャンセル] ボタンを押すと、メニューが表示され、ユーザーは取り消し方法として [破棄] と [下書きを保存] のいずれかを選択できます。

  注 コンテキストに依存するテキスト選択では、フライアウトではなくコンテキスト メニューを使います。
- **詳細情報の表示:** ユーザーが関心を持っている、画面上の項目に関するより詳しい情報を表示します。

  例: ブラウザーで、InPrivate ブラウズ中にユーザーが InPrivate アイコンを選択するとします。フライアウトが表示され、ユーザーに InPrivate モードに関するより詳しい情報を示します。多くの場合、ブラウザーの UI は簡素な状態ですが、関心を持ったユーザーから要求された場合は、詳細情報を提示できます。

- ユーザーが Tab キーを押したときに、フライアウト内の項目を移動する順序を指定します。必ず、タブ インデックスを正の値に設定し (0 は未設定と同じと見なされます)、フライアウト内のタブ インデックスの範囲がフライアウト外の要素のタブ インデックスの範囲と重複しないようにしてください。そうしないと、ユーザーはフライアウトとメイン アプリのページ間を移動することになります。
- ユーザーがフライアウトでコマンド ボタンを押さず、メニュー項目も選択していない場合は、フライアウトをプログラムを使って消さないようにします。たとえば、ユーザーが単に設定を切り替えただけの場合は、フライアウトを自動的に消しません。
- メッセージ、エラー、警告、その他の UI が、その時点でのユーザーの操作によって直接発生している場合以外、フライアウトは使いません。例: 更新プログラムが利用できる、試用版が期限切れになった、インターネットが使えない、などの通知

は、フライアウトでは表示しません。このような UI のサーフェスのガイダンスについては、「適切な UI サーフェスの選択: エラー」をご覧ください。

- エクスペリエンスが、手順の多い操作、複数の画面、多数の UI を必要とするものである場合、フライアウトは使いません。代わりに、アプリのキャンバスに UI を統合します。例:
  - ユーザーが、多くのテキスト入力を行うウィザードを操作している。
  - ユーザーが多数の設定を変更している。
- アプリの主要なコマンドには、フライアウトを使いません。それらには、アプリ バーを使います。詳しくは、「アプリ バーのガイドライン」をご覧ください。
- テキストを選択するためのコマンド専用のメニューが必要な場合、フライアウトは使いません。代わりに、コンテキスト メニューを使います。 「コンテキスト メニューのガイドライン」をご覧ください。
- その状況で必要がないコントロールはフライアウトに含めないようにします。たとえば、ユーザーが行うことができる操作がない場合は、ボタンを含めません。[閉じる] や [OK] ボタンは必要ありません。簡易非表示機能 (ユーザーがフライアウトの外側で画面上の任意の場所をタッチするとフライアウトが消える機能) によって適切に処理されます。同様に、必ずしも必要でないタイトルは含めません。
- フライアウト自体によって設定される余白以外の余白は追加しません。フライアウトは、コンテンツに合わせて、できる限り小さくします。
- 次のような理由で、フライアウトは状況と無関係な位置 (画面の中央など) には配置しません。
  - UI が、それを呼び出した操作と無関係な位置に表示されると、ユーザーはその UI を探す必要があり、操作に手間取ります。エクスペリエンス全体が中断され、使いにくく不安定な UI になります。
  - フライアウトが表示されたことにユーザーが気付かず、タップを続けたために、間違ってフライアウトを消す場合があります。そのような場合は、アプリの反応が悪いと思われます。
  - ユーザーは、中央 (またはその他の恣意的な場所) に表示されるウィンドウには [閉じる] または [キャンセル] ボタンがあると認識しているので、簡易非表示オプションが用意されていてもボタンを使おうとします。そのため、簡易非表示 UI が効果を発揮できません。

その他の使い方のガイダンス

フライアウトの各部

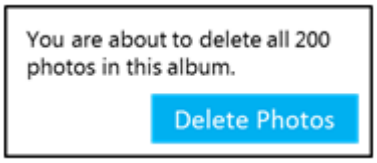
フライアウトには、3 つのコンポーネントがあります。それは、タイトル、メイン コンテンツ、コマンド ボタンです。

次の表は、フライアウトを使う一般的な状況で各コンポーネントをどのような場合に使えばよいかを示しています。

- **情報の収集:**

タイトル	なし
メインコンテンツ	必要なコントロールだけを含めます。説明や、"詳細情報" へのリンクなどは、最小限にします。ユーザーが設定を変えたり、オン/オフのスイッチを切り替えたりした場合は、できるだけ早く変更を確定します。カスタム コンテンツが操作された場合は、フライアウトを消さないようにします。コマンド ボタンがある場合以外は、ユーザーがフライアウトを手動で消せるようにします。
コントロール	ボタンが単にユーザーの変更を確定するためだけにある場合、そのボタンは不要であり、変更は自動的に確定されるようにする必要があります。ボタンが操作 (ログイン、ドキュメントの保存など) を開始する場合や、ユーザーが確定するテキストを入力した場合は、ボタンを使うのが適しており、フライアウトはユーザーがボタンを押したときに消えるようにします。ただし、ユーザーはフライアウトの簡易非表示機能によって、確定せずに取り消すことができます。 

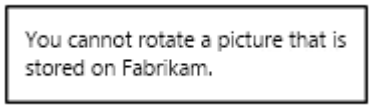
- **警告と確認:**

タイトル	なし
メインコンテンツ	操作を行う前に考慮する必要があるという警告をユーザーに表示します。質問形式の表現は使わないでください。 
コントロール	ユーザーが開始する操作 (削除など) だけを含めます。反対の操作や [キャンセル] ボタンは含めません。フライアウトを消すことで、そうした処理が行われます。

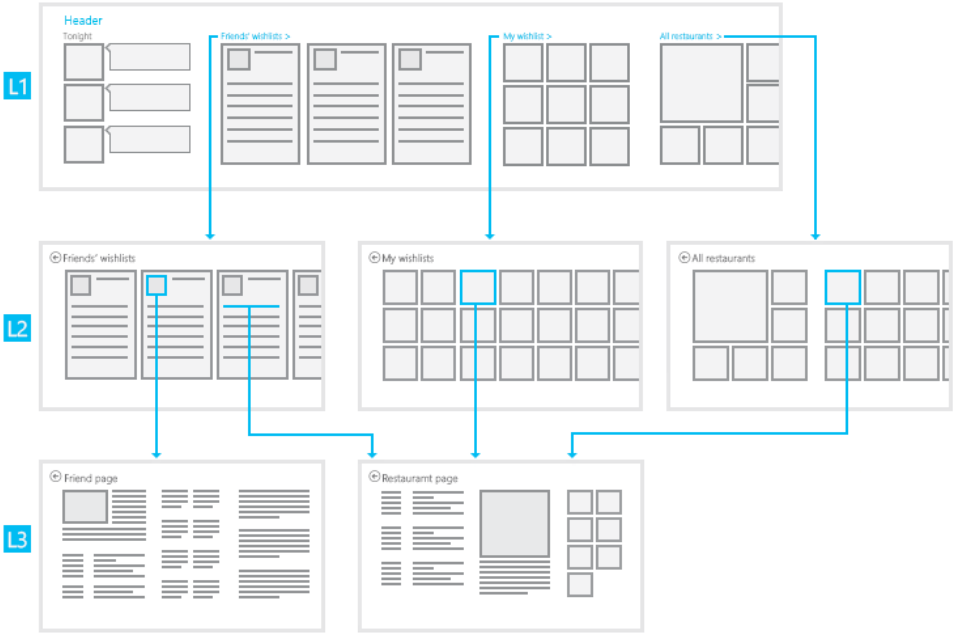
- **ドロップダウン メニュー:**

タイトル	なし
メインコンテンツ	ユーザーが操作できるメニュー項目のリストを含めます。 
コントロール	これらのボタンには MenuCommand を使います。

- **詳細情報の表示:**

タイトル	状態を説明するオプションのタイトルか、フライアウトを呼び出すために使われたアイコンの説明。 
メインコンテンツ	情報を含めます。 
コントロール	フライアウトの情報に対する操作を行うための、オプションのボタンを配置します。

ハブ コントロールのガイドライン (Windows ストア アプリ)



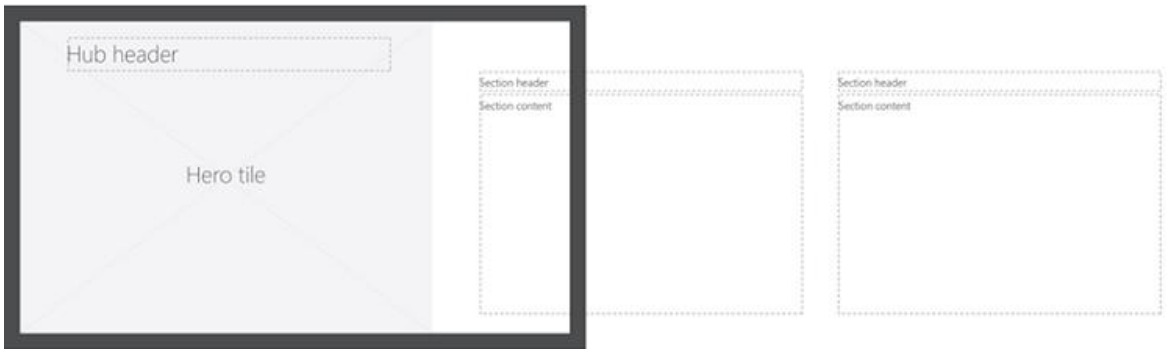
説明

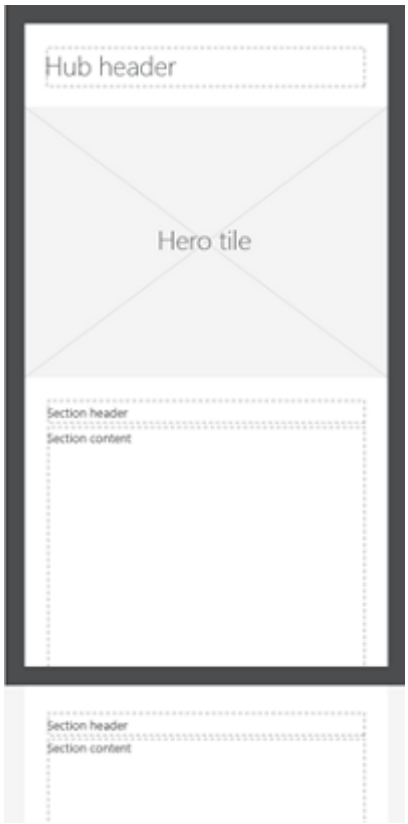
ハブ コントロールは、階層型ナビゲーション パターンを使って、リレーショナル情報アーキテクチャを必要とする Windows ストア アプリをサポートします。

このナビゲーション パターンが最も便利なのは、アプリのコンテンツを、詳細レベルが異なり、優先される参照順序のある、関連した別個のセクションやカテゴリに整理できる場合です。

このコントロールは、視覚的に優れたアプリを設計するための基盤になります。ハブ コントロールは、ブランドに合わせてアプリをカスタマイズする柔軟性とフレームワークを提供します。ハブ ページで多様なビジュアルを提供し、ランディング ページからセクション ページや詳細ページまで、ユーザーをアプリに誘導する必要があります。

最初のグループには、ヒーロー画像やコンテンツ セクションを使えます。



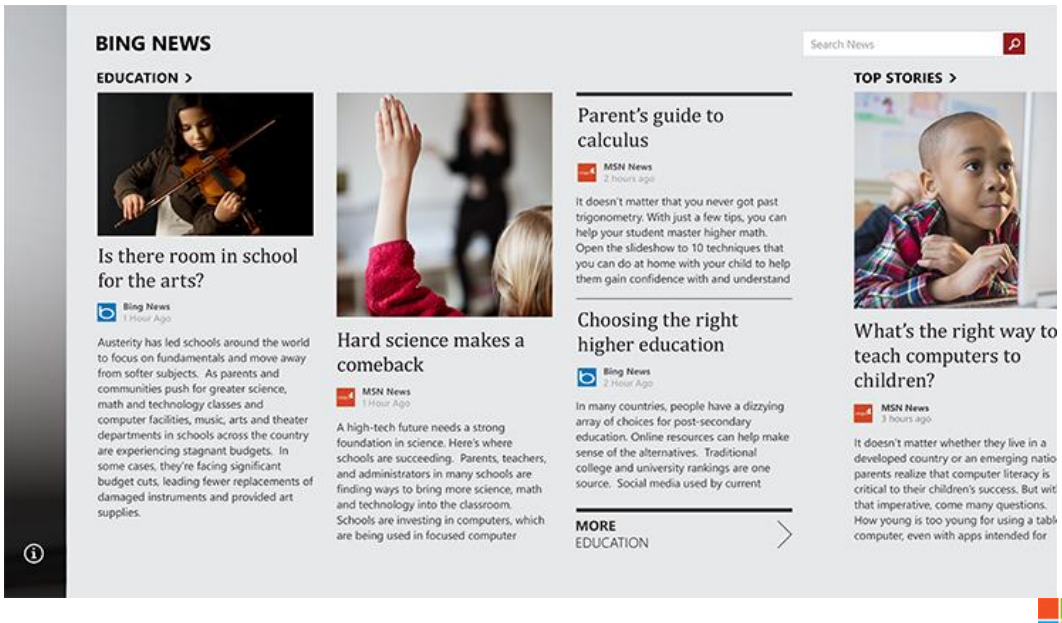


ヒーローには、最も強調したい内容を失うことなく垂直方向と水平方向にトリミングできる大きい画像を使ってください。

次の例は、1 つのヒーロー画像と、その画像を横長、縦長、狭い幅で表示するためにトリミングする方法を示しています。

例

ハブは、設計上の大きな柔軟性を備えています。 そのため、魅力的で視覚に訴えるさまざまなエクスペリエンスを提供するアプリを設計できます。 次のセクションでは、色、タイポグラフィ、画像、グラフィックス、構成を工夫して、さまざまな設計とスタイルの可能性を示すネイチャー アプリを示します。



ハブ テンプレート

ハブ テンプレートは基盤です。Microsoft Visual Studio のハブ テンプレートを使ったネイチャー アプリの例を示します。

ランディング ページ:

詳細ページ:

コンテンツはすっきりと整理されているものの、同じテンプレートを使った他のアプリと区別が付きません。 テンプレートはアプリの基盤として利用するにとどめ、それを基に個性のあるエクスペリエンスを構築する必要があります。

バリエーション 1 - モジュール

この設計は、大胆な色使いと画像のトリミングが特徴的なモジュール型レイアウトを使って定義されています。

ランディング ページ:

詳細ページ:

色　色のグラフィックス効果は、コンテンツを強化し、レイアウト全体にビジュアルな構造を与えます。この例では、クールな灰色と、インパクトのある大胆で鮮やかな色が使われています。 ランディング ページでは、1 つのセクションの背景色として明るい赤色が使われ、画像内にある同様のやや暗い赤色を補完しています。大胆な色使いと無彩色である灰色のバランスが取れており、内容を邪魔せず、補完するようにデザインされています。

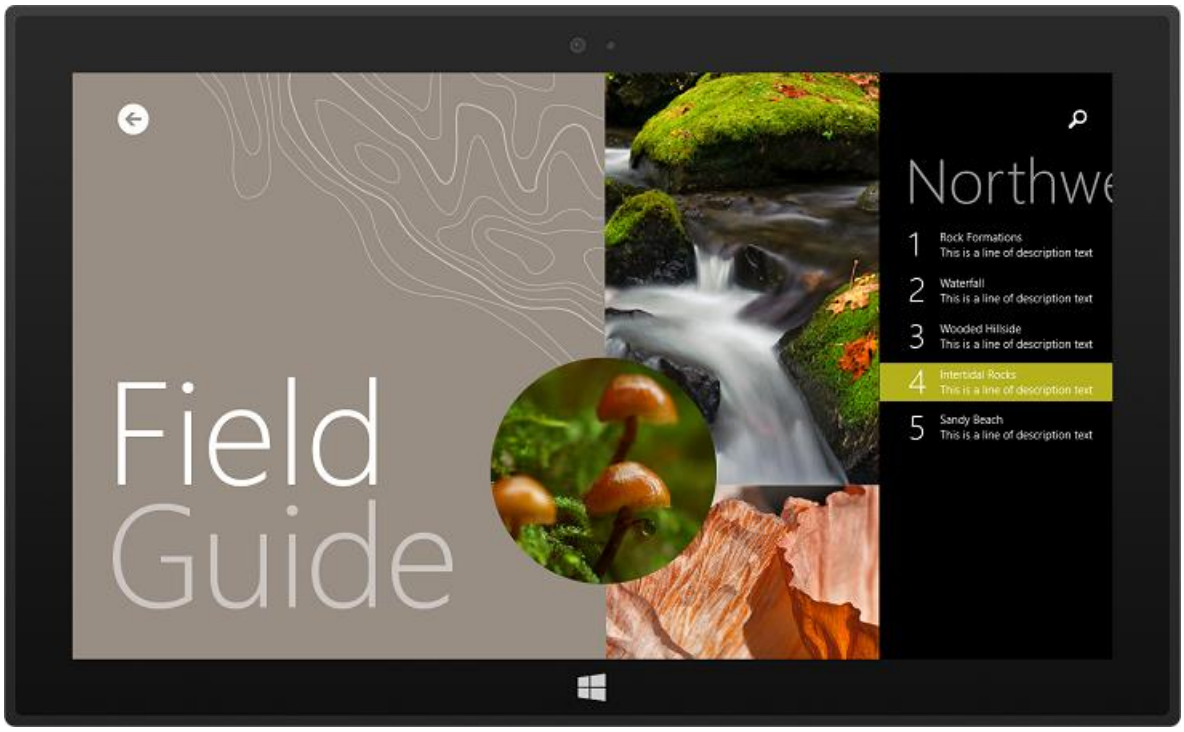
画像　写真はこのデザインの焦点です。 画像は大胆な色とグラフィックスのトリミングを考慮して慎重に選ばれました。 画像はフルブリードで (アプリの端に) 配置され、イマーシブな感覚を際立たせています。画像は、コンテンツをはっきり示し、コンテキストと意味が明確になるような形で使われます。ヒーロー画像がメインの場合、周りの画像は、メインの画像を邪魔せず、際立たせるような使い方ができます。

構成　ヒーロー画像は、サイズが大きく、ビジュアルに大きなインパクトがあるため、目立ちがちです。ヒーロー画像の右側にある各グループには、テキストと画像から成る個々のコンテンツがあります。構成と表示の向きでは、グループごとに異なる要件に対応したレイアウトが考慮されています。グループの順序は、アプリの全体的なコンテンツ階層と水平方向の構造をサポートしています。

タイポグラフィ　この例では、書体のサイズと色により、コンテンツ階層とインターフェイスの構造がサポートされています。サイズ、位置、スペースは、表示される情報の階層を基に論理的に定められます。ヒーロー画像がサイズと表示における例外であるのと同様に、ヒーローの書体も言語の要件に応じて例外になることがあります。ヒーローの右側の 3 つのグループは、データが中心のモデルです。タイトルの扱いに一貫性があり、ベースのグリッドに合わせて配置されていることから、結び付きが感じられます。

バリエーション 2 - "端から端まで" の縦方向ウィンドウ

この設計は、素朴さとモダンな美しさを併せ持つオーガニックな要素を配置した、縦方向ウィンドウの "端から端まで" の構成を示しています。



ランディング ページ:

詳細ページ:

色　緑色、黒色、茶色という素朴な色使いにより、自然の画像を補完しています。

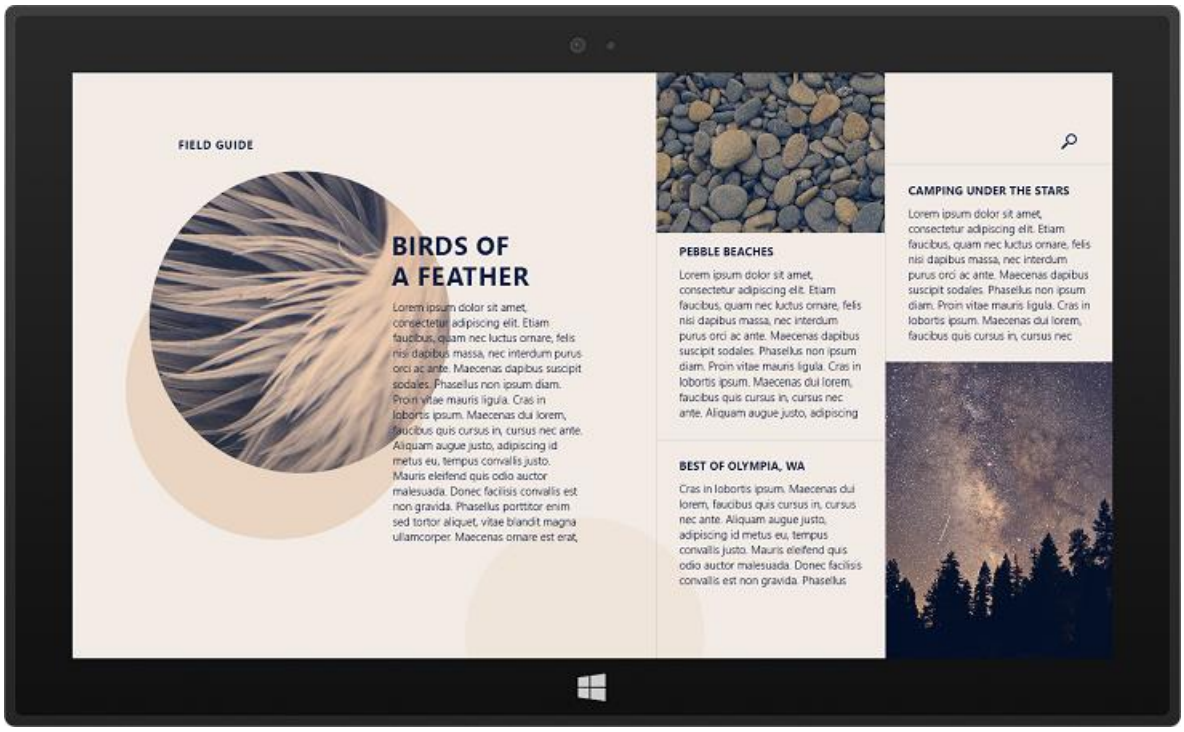
タイポグラフィ　通常の太さと細い Segoe UI をさまざまなサイズで使っています。 書体のスケールと透明度により、ヒーロー セクション、グループのタイトル、地図上の興味深いポイントを強調しています。

グラフィックス　ハブ ヒーロー セクションの等高線図のグラフィックスは、自然というテーマを強調し、アプリのグリッドによる構成とビジュアルな対照をなしています。

構成　このアプリは、テキストと画像から成るカラフルな縦方向ウィンドウで構成されています。 円形のアクセント画像、グループ ヘッダー、リード文などの要素が縦方向ウィンドウをまたいで配置されているため、構成のおもしろさが目立ち、主要なコンテンツ領域に目が行きます。

バリエーション 3 - 構造化された列

この設計は、柔らかい配色と画像で穏やかな印象を与える、構造化されたコラム式のレイアウトを採用しています。



ランディング ページ：

詳細ページ：

色　明るい中間色による柔らかな色使いにより、穏やかな印象を与えます。

画像　中間色またはセピア調の画像により、柔らかい配色を補完しています。色調の似た画像により、レイアウト全体に統一感があり、ビジュアルなノイズが減少しています。

グラフィックス　透明な円形のグラフィックスが、直線的な構成におもしろみを与えています。

構成　繊細な 1px の灰色の線でハブ内のグループが区切られています。 広い空白に対して小さいタイポグラフィを使うことで、このアプリの穏やかな魅力がさらに高まっています。

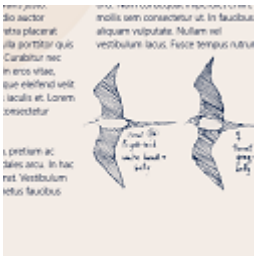


図　メディアの図やおもしろい書体が、テキストの大きなブロックの間に挿入されています。ここにあるスケッチが記事にテクスチャと個性を与えています。

バリエーション 4 - 傾斜

このバリエーションの特徴は画像を区切る傾斜であり、背面におもしろみを与えています。

ランディング ページ：

詳細ページ：

色　明るい色合いですが、傾斜したアクセントの図形には大胆な色を使っています。

タイポグラフィ　本文には Segoe UI を使っているものの、タイトルとグループ ヘッダーのテキストには対照的な Bebas Neue を使っています。 異なる書体を使うと、簡単かつ効果的にアプリに独特な印象を与えることができます。

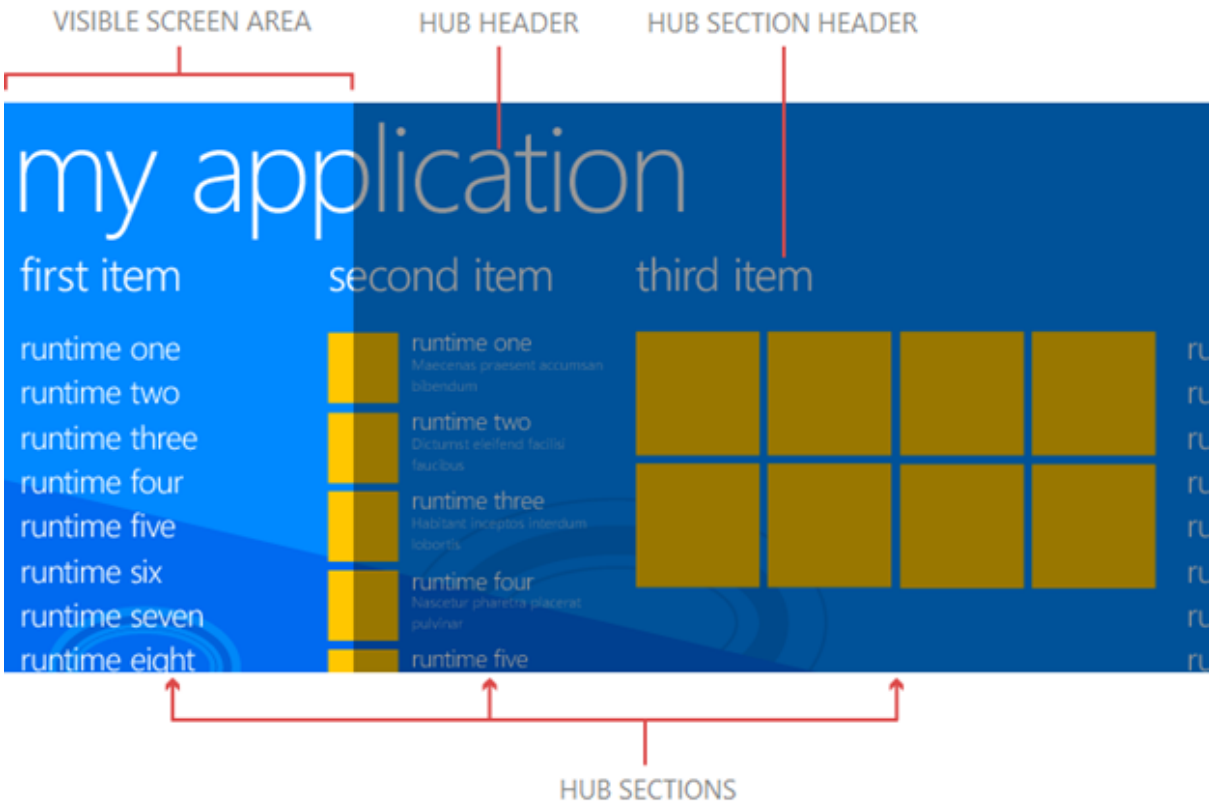
グラフィックス　ヒーロー画像の標準的なハブ構成と、リストを含む個別のグループを使っています。 ただし、ヒーロー画像と背景画像の両方に傾斜したグラフィックス要素を追加することで、このデザインに動きと個性が生まれています。

推奨と非推奨

- 対話型ヘッダーを使い、山形のグリフ (>) をヘッダー テキストの末尾に置くことで、さらにコンテンツがあることを示します。詳しくは、「コントロール (XAML と C#、C++ または VB)」の「**ハブ**」または「コントロール (HTML と JavaScript)」の「**ハブ**」をご覧ください。
- ハブ内にハブを作成しないでください。別のセクションまたはページに移動できるように対話型ヘッダーを使います。
- グループ内でコンテンツを動的に再配置し、さまざまなウィンドウ サイズに対応できるようにします。詳しくは、「ウィンドウ サイズのガイドライン」をご覧ください。

- 多くのセクションがある場合は、ハブにセマンティック ズームを追加することを検討します。これには、アプリが狭い幅に合わせてサイズ変更されたときにセクションを見つけやすくなるという利点もあります。
- メイン アプリ コンテンツには、重ねて表示されるテキストに調和した画像を慎重に選びます。
- ハブ テンプレートを基盤として使って、アプリの特長を反映したレイアウトになるようにカスタマイズします。ハブ コントロールの次の機能をカスタマイズできます。
    - セクションの数
    - 各セクションのコンテンツの種類
    - セクションの配置と順序
    - セクションのサイズ
    - セクションとセクションの間隔
    - セクションとハブの上端または下端の間隔
    - ヘッダーとコンテンツのテキストのスタイルとサイズ
    - 背景、セクション、セクション ヘッダー、セクション コンテンツの色
- 水平方向にパンするハブ内では垂直方向にパンするセクションを使わないでください。スワイプによる選択とマウスのスクロールが適切に機能しません。詳しくは、「パンのガイドライン」をご覧ください。

ハブ コントロールのガイドライン(Windows Phone)



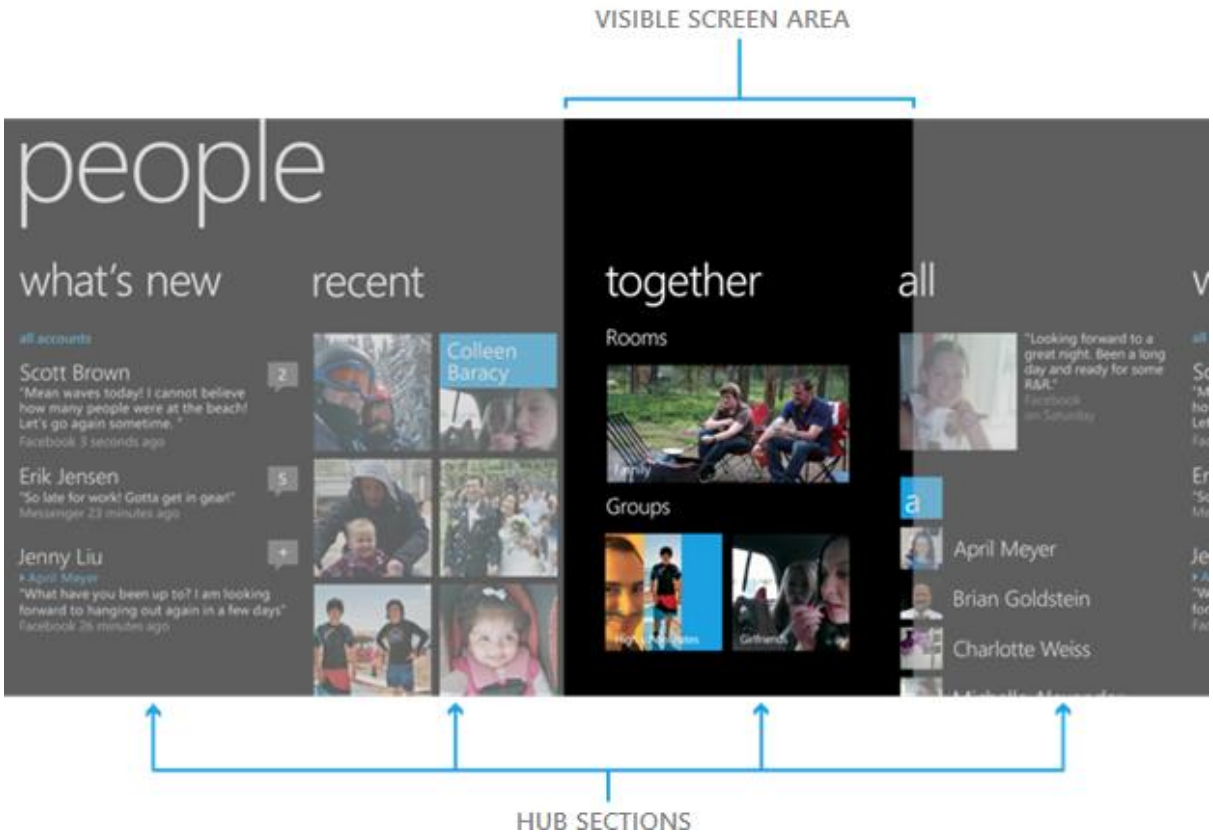
説明

縦向きでのみ使用が想定されている電話のハブ コントロールには、左右に動かすごとができる一連のセクションが表示されます。これは、アプリの全画面コンテナーであり、ナビゲーション モデルでもあります。

ハブ (以前のパノラマ) エクスペリエンスは、Windows Phone 本来の外観に含まれます。電話画面の範囲内に収まるように設計されたアプリとは異なり、ハブ アプリでは、画面の範囲を超えた横長の仮想キャンバスを使ってコントロール、データ、サービスを表示する独自の方法が用意されています。Windows Phone のこの本質的に動的なビューでは、レイヤー化されたアニメーションとコンテンツを使って、視差効果と同様に、レイヤーがさまざまな速度でスムーズにパンします。

ハブ アプリのセクションは、さらに詳しい内容を表示するための出発点としての役割を果たします。目標は、視覚に訴えるコンテンツをユーザーに表示することです。

例



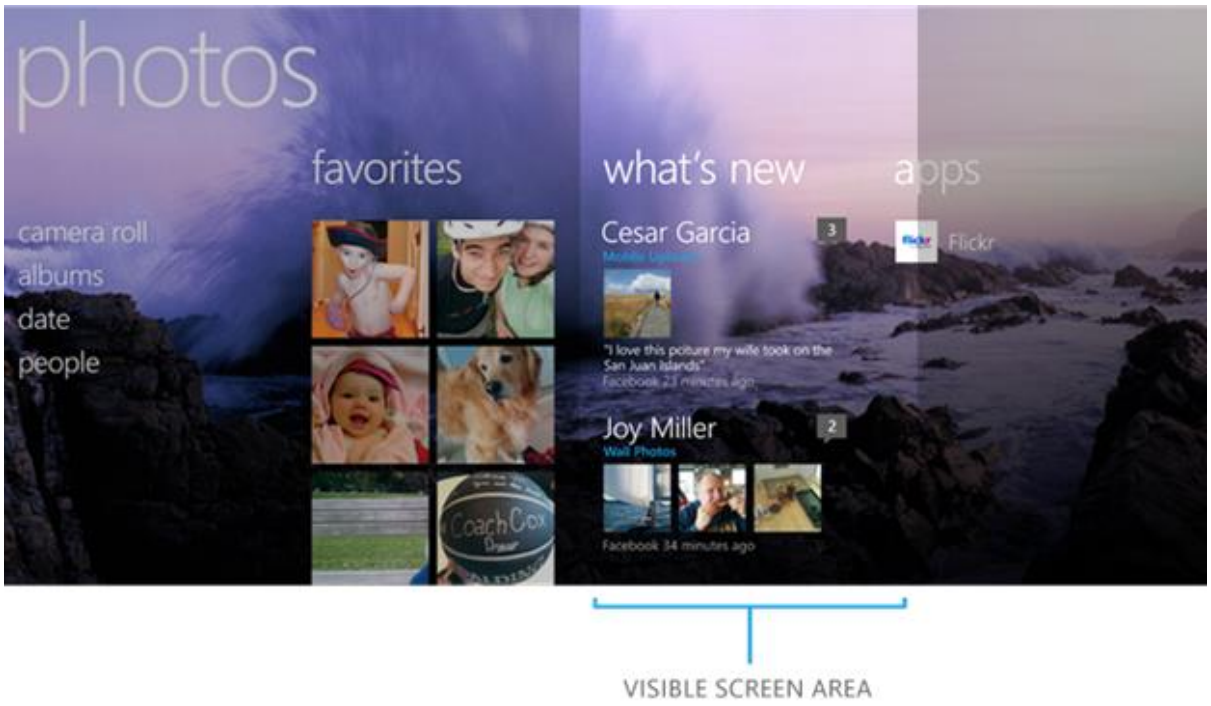
ユーザー インターフェイスは、背景色や背景画像、ハブ ヘッダー、ハブ セクション ヘッダー、ハブ セクションなど、それぞれ単独で移動するレイヤーから構成されます。

設定した場合、背景画像は一番下のレイヤーで、ハブに高級感のある雑誌のような印象を与えます。背景画像は、通常、フルブリード画像が使われ、アプリの中で最も視覚的に訴える部分となる場合もあります。

ハブ ヘッダーは、アプリを識別するもので、ユーザーがどのような方法でアプリを開始した場合でも目に入るようにする必要があります。

ハブ セクションは、ハブ アプリのコンポーネントで、他のコントロールやコンテンツを格納しています。ハブ セクションは、タッチによるパンやフリックと同じ速度で移動します。ハブ セクション ヘッダーは、どのハブ セクションでも省略可能です。

サムネイルは重要な要素になる場合があり、他のページのコンテンツやメディアにリンクしています。



使い方のガイダンス

アプリの要件に基づいて、ハブ コントロールに複数のハブ セクションを追加し、各セクションに異なる機能目的を持たせることができます。たとえば、あるセクションには一連のリンクとコントロールを含め、別のセクションはサムネイル画像のリポジトリとして使います。ユーザーは、ハブ コントロールに組み込まれているジェスチャのサポートを使って、これらのセクションを前後に動かすことができます。

設計のガイダンス

- ハブ コントロールの動作とレンダリングは、縦向きでのみサポートされます。
- ハブ コントロールにはテーマを設定できるため、既定の色などを上書きすることができます。
- Windows Phone 用のハブ コントロールの折り返し効果を使うと、最後のセクションから最初のセクションに折り返すことができ、その逆も可能です。1 つまたは 2 つのセクションを含むハブでは折り返し表示されません。3 つ以上のセクションを含むハブでは折り返し表示されます。

- Windows Phone では、ハブでアプリ バーを使う場合、アプリ バーのモードを最小に設定します。このモードは、ハブ ページの画面領域を最大限利用できるように設計されています。詳しくは、「Windows Phone のアプリ バー」をご覧ください。
- ハブ コントロールの起動中には、システム トレイの進行状況バーを表示するか、全画面の "読み込み中" インジケーターを表示します。
- ハブ コントロールのセクションが更新中でもユーザーの操作をブロックしていない場合に、システム トレイの進行状況バーを表示します。
- 初回アクセス時: 最初に表示されるセクションでは、ハブ ヘッダーが左側に正しく整列されています。どのセクションを既定にするかに関する標準のガイダンスはありません。表示するコンテンツによって異なります。
- 同じハブ コントロールに対する 2 回目以降のアクセス時には、ユーザーは中断したセクションに戻されます。
- ハブ コントロールに 5 個を超えるセクションを作らないでください。これは、パフォーマンス上の理由によります。また、ユーザーが移動する必要のある領域の数を制限するためでもあります。コンテンツが複雑なほど、使うセクションを少なくします。セクションが多すぎるとユーザーが混乱します。4 ～ 5 個のセクションであれば、ユーザーは現在のセクションと左右のセクションを把握できます。
- ハブ コントロール内ではピボット コントロールを使わないでください。逆の場合も同じです。ただし、ハブ セクション内の項目を、ピボット コントロールが含まれる別のページにリンクすることはできます。
- ハブ コントロール内でパンやスクロールできるコントロールを使わないでください。たとえば、ハブ セクション内にマップ コントロールを配置すると、ハブ コントロールが使いにくくなる場合があります。ジェスチャ入力が混同されます。たとえば、ハブ コントロールのセクション内にあるスライダーを左にスライドしようとすると、セクションをスクロールしたいのか、スライダーを動かしたいのかがはっきりしません。これを解決するには、ジェスチャ入力を必要とするコントロールを個別のページに配置し、そこにナビゲートします。ジェスチャが無効になっているコントロール (おそらくはマップ) はハブ セクションに配置しても問題ありません。マップをアクティブ化するボタンをオーバーレイできます。このボタンを押すと、マップだけが含まれる別のページに移動します。その後、ユーザーは、戻るボタンを押してハブ セクションに戻ることができます。

- ハブ コントロールとピボット コントロールの使い分けについて詳しくは、以下のトピックをご覧ください。
  - Windows Phone のホーム ページ メニュー (Panorama コントロールまたは Pivot コントロール) を持つセントラル アプリ ハブ
  - Windows Phone のパネル領域 (Panorama コントロール) を持つセントラル アプリ ハブ
  - Windows Phone のアプリ タブ (Pivot コントロール)

ハブ ヘッダー:

- プレーンテキストまたは画像 (ロゴなど) を使います。ロゴとテキスト (またはその他の UI 要素) など、複数の要素を使うこともできます。
- ヘッダーのフォントや画像の色には背景画像のどこでもはっきりと見えるようなフォントや色を使います (この 2 つは個別に移動するため)。別のフォント、サイズ、色を使う特別なブランド戦略の必要がない限り、システム フォントとスタイルを使ってください。
- ヘッダーをアニメーション化したり、そのサイズを動的に変更したりしないでください。
- 一貫性を保つため、ハブ ヘッダーをスタート画面の起動タイルを同じにします。
- ハブ コントロールのレイアウト時やヘッダーの設計時は、システム トレイやその他の要素がふさがれないようにしてください。

ハブ セクション ヘッダー:

- プレーンテキストまたは画像 (ロゴなど) を使います。ロゴとテキスト (またはその他の UI 要素) など、複数の要素を使うこともできます。
- ヘッダーのフォントや画像の色には背景画像のどこでもはっきりと見えるようなフォントや色を使います (この 2 つは個別に移動するため)。別のフォント、サイズ、色を使う特別なブランド戦略の必要がない限り、システム フォントとスタイルを使ってください。
- ヘッダーをアニメーション化したり、そのサイズを動的に変更したりしないでください。

ハブ セクション:

- セクションのコンテンツに含まれるテキストや画像を吟味することによって、ハブ コントロール エクスペリエンスの美しさを保ち、ハブに圧迫感やまとまりのなさが生まれないようにします。
- 垂直スクロールを使う場合は向きを考慮します。ハブ セクションでの垂直スクロールは、セクションの幅が画面の幅より広い間、問題ありません。
- コンテンツを表示するまではハブ セクションを非表示にすることを検討してください。

背景画像

- 暗く、柔らかで、低コントラストの背景が最適です。単色、またはグラデーション。
- 繊細で控えめな写真を背景に使うと、ハブ コントロールの視覚的効果を高めることができます。一方、明るい色が多く含まれる写真は、セクションが見づらくなる場合があるため、使わないようにします。実用的な方法の 1 つとして、写真の上に半透明の黒または白のフィルター (四角形) を使います。ビットマップ編集ツールでこの処理を実行し、画像の色を抑えることができます。
- 背景画像はハブ コントロール全体にまたがる必要があります。つまり、背景画像の縦横比とハブ コントロールの縦横比を一致させ、画像のサイズは最も一般的なデバイス解像度、最大のデバイスの解像度、パフォーマンスを考慮したものにする必要があります。ファイルのサイズを抑えるには、JPEG 形式をお勧めします。
- 背景画像は、アプリの実行中でも切り替えることができます。ただし、同時に表示できる画像は 1 つだけです。

サムネイル

- 画像全体を縮小するのではなく、内容を判断可能な被写体を際立たせるようにトリミングした画像を使います。テキストがないと画像の内容が明確でない場合は、コンテンツの説明のためにテキストを 2 行まで使用できます。

ラベルのガイドライン

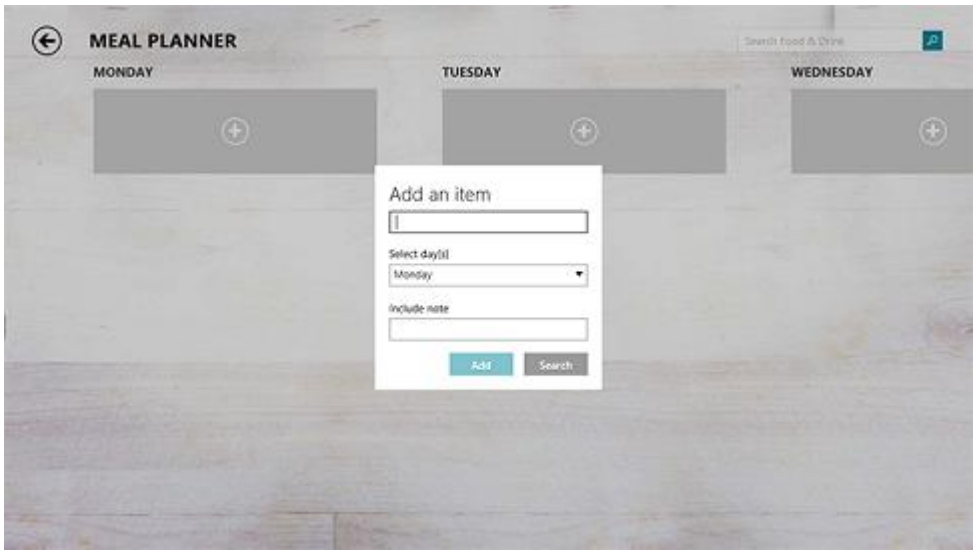
説明

ラベルは、コントロールまたは関連するコントロールのグループの名前やタイトルです。

XAML では、多くのコントロールに組み込みの Header プロパティがあり、これを使ってラベルを表示します。Header プロパティがないコントロールの場合、またはコントロールのグループにラベルを付ける場合は、代わりに **TextBlock** を使います。

HTML では、**label element** を使います。

例



推奨と非推奨

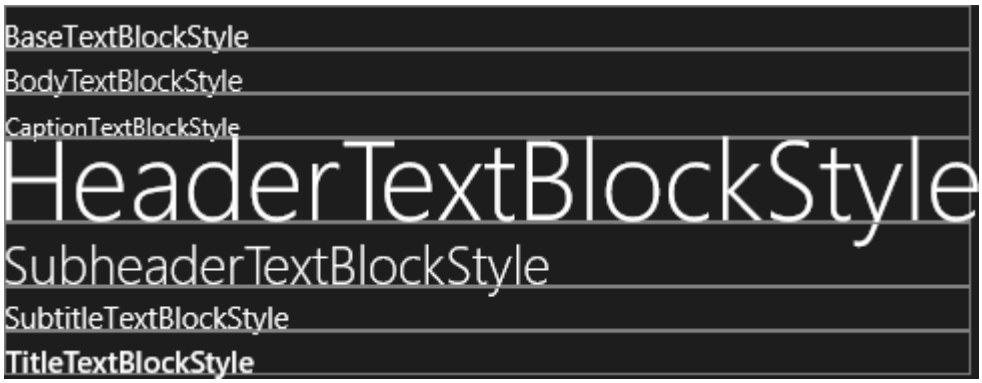
- 隣接するコントロールに入力する必要がある内容が明確ではない場合には、ユーザーへの説明のためにラベルを使います。たとえば、テキスト入力ボックスの上に "名前" を表示します。また、関連するコントロールのグループにラベルを付けることや、関連するコントロールのグループの近くに説明テキストを表示することができます。

- コントロールにラベルを付ける場合、説明テキストの文ではなく、名詞や簡潔な名詞句のラベルを入力します。コロン、その他の句読点は使わないでください。
- 説明テキストの長さにはそれほど制限はありません。句読点を使うこともできます。

Windows ストア アプリ用の XAML スタイル ギャラリー

組み込みアプリのようなスタイルのアプリを作るには、Windows ストア アプリでこの XAML スニペットを使って、TextBlock スタイル ギャラリーを参照することが役立ちます。

```
<StackPanel>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="BaseTextBlockStyle" Style="{StaticResource BaseTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="BodyTextBlockStyle" Style="{StaticResource
BodyTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="CaptionTextBlockStyle" Style="{StaticResource
CaptionTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="HeaderTextBlockStyle" Style="{StaticResource
HeaderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="SubheaderTextBlockStyle" Style="{StaticResource
SubheaderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="SubtitleTextBlockStyle" Style="{StaticResource
SubtitleTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="TitleTextBlockStyle" Style="{StaticResource TitleTextBlockStyle}"/>
    </Border>
</StackPanel>
```



Windows Phone ストア アプリ用の XAML スタイル ギャラリー

組み込みアプリのようなスタイルのアプリを作るには、Windows Phone ストア アプリでこの XAML スニペットを使って、TextBlock スタイル ギャラリーを参照することが役立ちます。

```
<StackPanel>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="BaseTextBlockStyle" Style="{StaticResource BaseTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="BodyTextBlockStyle" Style="{StaticResource
BodyTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ComboBoxPlaceholderTextBlockStyle" Style="{StaticResource
ComboBoxPlaceholderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ControlContextualInfoTextBlockStyle" Style="{StaticResource
ControlContextualInfoTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ControlHeaderTextBlockStyle" Style="{StaticResource
ControlHeaderTextBlockStyle}"/>
    </Border>
```

```
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="FlyoutPickerTitleTextBlockStyle" Style="{StaticResource
FlyoutPickerTitleTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="GroupHeaderTextBlockStyle" Style="{StaticResource
GroupHeaderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="HeaderTextBlockStyle" Style="{StaticResource
HeaderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ListViewEmptyStaticTextBlockStyle" Style="{StaticResource
ListViewEmptyStaticTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ListViewItemContentTextBlockStyle" Style="{StaticResource
ListViewItemContentTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ListViewItemSubheaderTextBlockStyle" Style="{StaticResource
ListViewItemSubheaderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="ListViewItemTextBlockStyle" Style="{StaticResource
ListViewItemTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="MessageDialogContentStyle" Style="{StaticResource
MessageDialogContentStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="MessageDialogTitleStyle" Style="{StaticResource
MessageDialogTitleStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="SubheaderTextBlockStyle" Style="{StaticResource
SubheaderTextBlockStyle}"/>
    </Border>
    <Border BorderBrush="Gray" BorderThickness="1">
        <TextBlock Text="TitleTextBlockStyle" Style="{StaticResource TitleTextBlockStyle}"/>
    </Border>
</StackPanel>
```

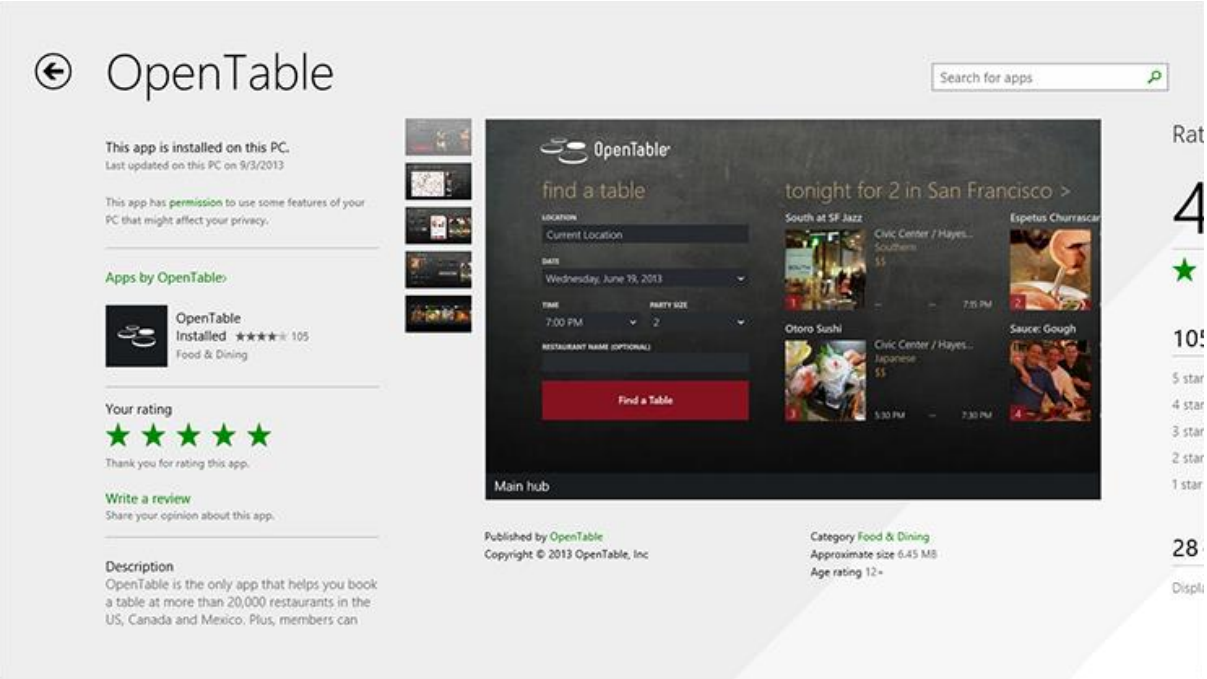
リンクのガイドライン

A hyperlink lets you give users a visual hint that certain text links to other content. Read more on the Dev Center. Text in a hyperlink element is treated like the rest of the text, so it participates in line-breaking.

説明

ハイパーリンクは、インライン テキスト (XAML と HTML) またはハイパーリンク ボタン (XAML) の形にすることができます。どちらの場合でも、ハイパーリンクは、ユーザーがタップすると、ブラウザーに Web ページが表示されたり、現在のアプリの別のページや同じページーの別セクションに移動したりーできるテキストです。

例



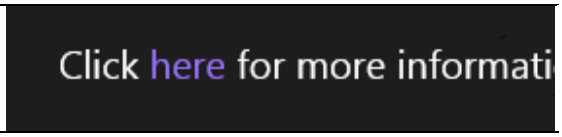
適切なコントロールの選択

ハイパーリンクは、ハイパーテキストが適切な対話操作である場合、つまりユーザーがテキストをタップしたときに、そのテキストに関連する詳しい情報を表示する場所に移動するときに使います。ー移動先は、**NavigateUri** プロパティ (XAML) または **href** プロパティ (HTML) の中に Uniform Resource Identifier (URI) でエンコードします。

動的テキストの中にインラインで表示されるハイパーリンクは、ハイパーリンク内で自動改行を実行できるため、**Hyperlink** (XAML) または **a element | a object** (HTML) を使います。改行する必要がない場合、またはハイパーリンクの視覚効果をカスタマイズする必要がある場合は、**HyperlinkButton** (XAML) を使います。

既定では、ハイパーリンクは従来の Web のハイパーリンクと同じように表示され、同じように動作します (区別できる色のテキストまたは画像で表示され、タップすると別の場所に移動できます)。ただし、**HyperlinkButton** (XAML) を構成する視覚効果をカスタマイズできます。ユーザーがハイパーリンクを操作すると、ハイパーリンクは状態 (外観) を変更することでフィードバックを返します。ハイパーリンクの状態には、Normal、Pressed、Disabled などがあります。

次に示すように、文中のボタンは見直しのチャンスです。ボタンをクリックすると関連する詳しい情報のある場所に移動する場合は、ハイパーリンクを使うことを検討してください。ボタンをクリックすると別の処理が開始される場合は、ボタンを別の場所にレイアウトし、テキストとボタンのコンテンツを作り直すことを検討してください。

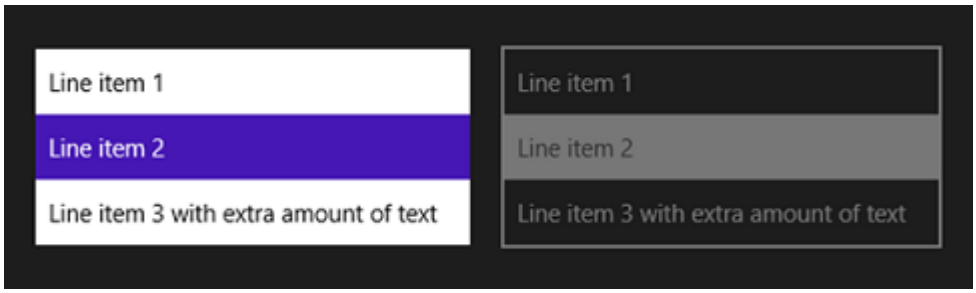
不適切	**適切**
	
この例では、ボタンはスペースを取りすぎることと場違いであるように感じられるため、別のコンテンツを表示するために使う適切なコントロールではありません。	この例では、ハイパーリンクは文中の他のテキストと同じように表示されるため違和感がありません。

推奨と非推奨

- ハイパーリンクの間に十分な間隔を取って、ユーザーが正確にタップできるようにします。
- ハイパーリンクの無効状態は、その状態が一時的である場合 ―(他のシステムによる処理が発生している場合など)― またはユーザーがそのハイパーリンクを有効にできる場合のみ使います。

- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- すべてのリンクにツールチップを設定します。これで、指でリンクが見えない場合でも、ユーザーはそのリンクで何が行われるかを確認できます。
- 外部サイトに移動する場合は、ツールチップ内にドメイン名を入れ、補助的なフォント色を使ってドメイン名のスタイルを指定します。ツールチップにドメイン名を追加すると、ユーザーは外部サイトに移動しようとしていることがわかるため、リンクをクリックしたときに驚くことがなくなります。トップレベルのドメインを表示するだけで十分です。
- リンクに訪問したことがあるかどうかをユーザーが気にしない場合は、そのページに訪問済みかどうかにかかわらずリンクの外観が同じになるようにそのリンクの visited 状態スタイルを指定します。訪問済みリンクの既定のスタイルでは、その外観は訪問されたことがないリンクと異なっています。リンクが訪問済みであるかどうかをユーザーが気にしないこともあります。リンクがアプリのメイン ナビゲーションの一部である場合などがそうです。
- リンク テキストは簡潔なものにします。追加情報を提供する場合は、リンクのツールチップの中に配置します。
- リンクは、移動以外の処理で使わないでください。

リスト ボックス (または選択)のガイドライン



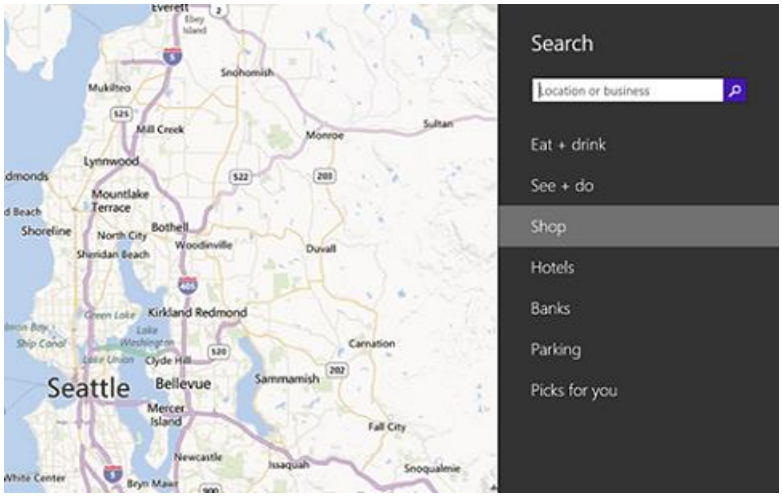
Windows アプリ：有効なリスト ボックスと無効なリスト ボックス

Windows Phone アプリ：リスト ボックス

説明

リスト ボックス (選択とも呼ばれます) では、ユーザーが通常は 1 つ、ただし時に複数の項目を項目のリストから選択する手段が提供されます。リスト ボックスの項目は、すべての項目を表示する領域がない場合には、スクロールできます。

例



適切なコントロールの選択

目立つ表示に値する重要性が項目にある場合は、リスト ボックスを使います。 リスト ボックスでは、重要な選択で完全な代替セットにユーザーの注意を向ける必要があります。可能な場合は、項目のリストのパンまたはスクロールが必要にならないように、リスト ボックスのサイズを設定します。リスト ボックスでは、3 ～ 9 個の項目が最適です。またリスト ボックスは、項目が動的に可変である場合に適しています。

次に、リスト ボックスの使用が適さないことを示す要因をいくつか示します。

- 項目が非常に少数である場合。常に同じ 2 つのオプションを持つ単一選択リスト ボックスは、ラジオ ボタンとして提示した方が適切である場合があります。3 つ、さらには 4 つの静的な項目の場合にも、ラジオ ボタンを考慮してください。
- リスト ボックスが単一選択であり、常に同じ 2 つのオプションを持ち、その一方が "でない" を意味するか、"オン" と "オフ" に対応する場合。単一のチェック ボックスまたはトグル スイッチを使います。
- 項目が非常に多数である場合。長いリストに適している選択は、グリッド ビューおよびリスト ビューです。非常に長いグループ化されたデータのリストでは、セマンティック ズームが推奨されます。
- 項目が連続する数値である場合。スライダーの使用を考慮します。
- 選択項目がアプリのフローで二次的な重要性しか持たないか、または大半の状況で大半のユーザーに既定のオプションが推奨される場合。ドロップダウン リストを使います。

推奨と非推奨

- リスト ボックスの目的、および現在選択されている項目が明確であることを確認します。
- タッチ フィードバックおよび項目の選択状態のビジュアル効果とアニメーションを予約します。

- リスト ボックス項目のテキストのコンテンツは、単一行に制限します。項目がビジュアルである場合、サイズをカスタマイズできます。項目に複数行のテキストかイメージが含まれる場合は、代わりにグリッド ビューかリスト ビューを使います。
- リスト ボックスが自動的にサイズ設定され、項目が動的である場合、リスト ボックスのサイズ変更がどのように行われ、またその周囲のビジュアルがどのように処理されるかを考慮します。動的な項目を含む固定サイズのリスト ボックスではサイズは変更されませんが、スクロールできます。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- 関連するオプションをグループ化する、最も一般的なオプションを最初に配置する、アルファベット順を使うなど、論理的な順序でリスト ボックス内の項目を並べ替えます。名前はアルファベット順、数値は数値順、日付は時系列順に並べ替えます。
- スクロール バーがコンテンツと重ならないように、コンテンツの右側に 27 ピクセルのパディングを追加します。
- コマンドの実行または他のコントロールの動的な表示と非表示の切り替えのために リスト ボックスを使わないでください。

その他の使い方のガイダンス

外観と操作

(ドロップダウン リストとは対照的に) リスト ボックスは常に開いています。テキスト文字列または数値の項目を含めることができ、他の任意のビジュアルにカスタマイズできます。創造的であることが推奨されます。アプリのコンテンツを直接操作できれば、ユーザーが理解しやすくなります。したがって、提示する実際のコンテンツ (おそらく製品の図面、写真) を示すこともできます。タップ時に項目でビジュアル フィードバックを常に提供し、選択時の項目が明確であることが必要です。

リスト ボックス コントロールでは、既定で項目のリストが垂直方向に提示されます。特に項目がグラフィックスや写真である場合に、項目を水平方向に提示するときは、リスト ボックス項目のレイアウトをカスタマイズできます。水平方向か垂直方向の折り返しグリッド

で項目をレイアウトしたい場合には、そうすることができます。また、既定で項目をそのようにレイアウトするグリッド ビューのみを使います。

ユーザーが項目を操作すると、項目は状態 (外観) を変更することでフィードバックを返します。リスト ボックスの項目の状態には、標準、押された状態、選択されていない状態、選択された状態、無効な状態などがあります。

項目をタップすると選択されます。複数選択モードでは、選択した項目をタップすると解除されます。単一選択モードでは、別の項目をタップするとその項目に選択が移動します。指で垂直方向にスワイプすると、項目のリストが慣性を伴って上下にスクロールします。リスト ボックスにはスクロール バーがあり、スクロール バーの位置が項目リスト内でのユーザーの相対位置、サイズが表示項目の比率を示します。スクロール バーは、スクロール中にのみ表示されます。

リスト ビュー と グリッド コントロールのガイドライン

説明

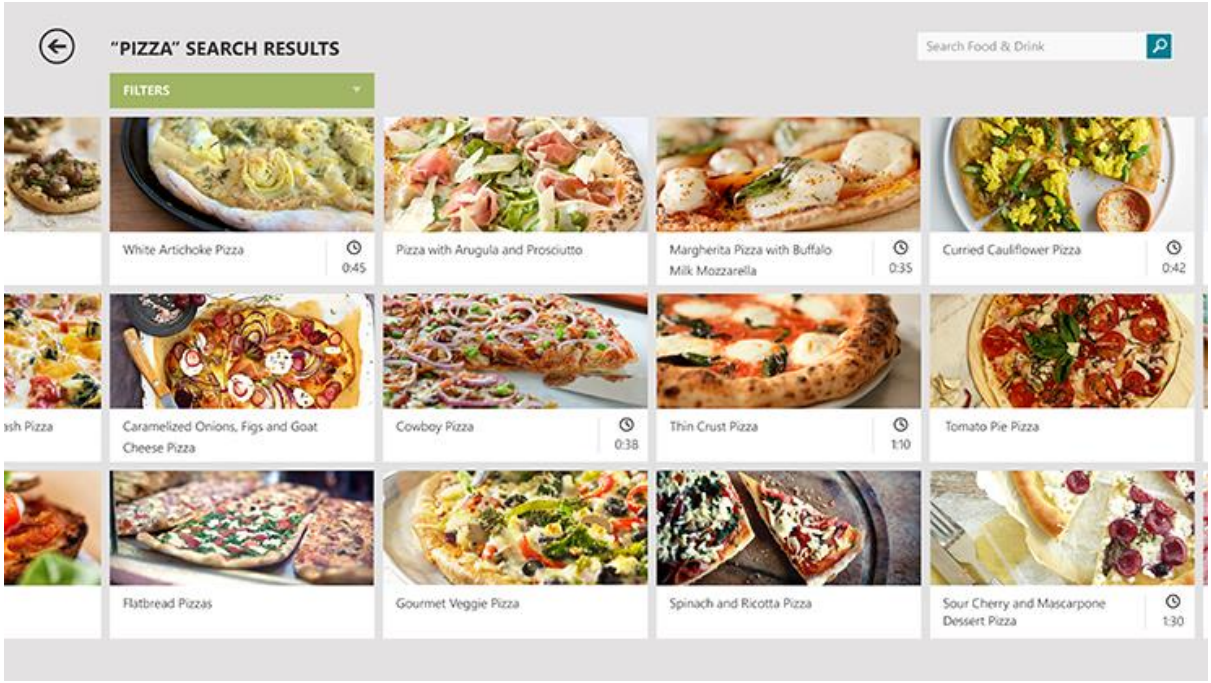
グリッド ビューまたはリスト ビューは、Windows ストア アプリ内のコンテンツのコレクションです。グリッド ビューとリスト ビューを使うことで、タッチ用に最適化された、ユーザーに対して一貫性のある項目表示エクスペリエンスが実現します。

これらのコントロールには、グループ化されていないグリッド、グループ化されたグリッド、リスト (HTML)、グループ化されていないリスト (XAML)、グループ化されたリスト (XAML) の複数の形式があります。リストは縦方向にパンします。グリッドは横方向にパンします。リストの読み取り順序は上から下です。グリッドの読み取り順序は上から下、左から右です。

注 JavaScript 用 Windows ライブラリ (WinJS) には、リスト レイアウトとグリッド レイアウトのオプションを備えた **ListView** コントロールが 1 つあります。C#/VB/C++ と XAML を使ったアプリでは、**ListView** と **GridView** という 2 つの別個のコントロールを使ってリスト レイアウトとグリッド レイアウトを作成します。次のガイドラインは、これらすべての種類のコントロールに適用されます。

例

標準的なグリッド ビューに表示された検索結果の一覧:



適切なコントロールの選択

データのコレクションを一連の項目としてユーザーに表示するには、リスト ビュー コントロールまたはグリッド ビュー コントロールを使います。 たとえば、リスト ビューを使って、メールの一覧、ショッピング カートの中の品目、画像の一覧、検索結果を表示することができます。

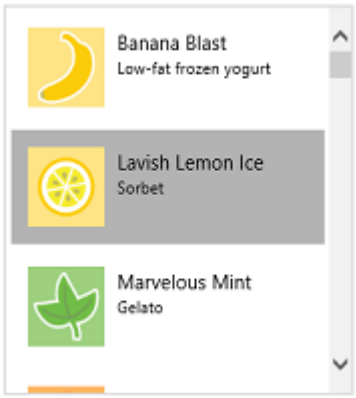
このコントロールは、次のような目的で使います。

- データをグループとして表示する
- ユーザーが 1 つ以上の項目を選べるようにする
- データの読み込み方法をカスタマイズする
- キーボード操作と選択方法を改善する
- 項目の一覧を動的に編集し組み込みアニメーションを取得する
- テンプレートを使って項目をカスタマイズするか、項目に独自のイベントを設定する
- さまざまなサイズの項目を表示する

リストとグリッドの使い分け

WinJS には、**ListView** の基本的なレイアウトとして、リスト レイアウトとグリッド レイアウトの 2 種類が用意されています (独自のカスタム レイアウトを作成することもできます)。C#/VB/C++ と XAML を使っている場合は、**ListView** と **GridView** という 2 つの別個のコントロールを使って同じ機能を実現できます。

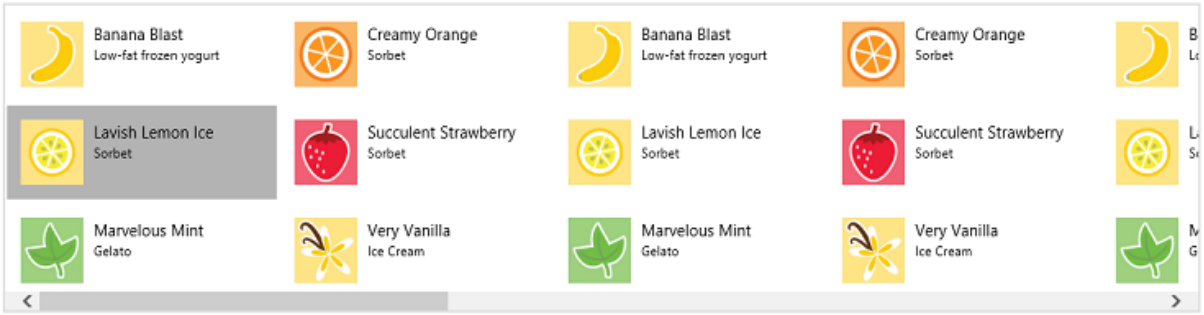
リスト: リスト レイアウトでは 1 つの列が表示され、読む順序は上から下で、パンやスクロールは縦方向に行います。このレイアウトではグループ ヘッダーやグループの境界などのグループ情報が表示されません。



リストは、次のような場合に使います。

- 幅の狭いウィンドウまたは縦方向にコンテンツを表示する。
- 領域を節約する必要がある (リスト レイアウトはグリッド レイアウトよりも使用領域が少なくて済みます)。
- 項目をグループ化する必要がない。
- マスター/詳細ビューにマスター ウィンドウを作成する。マスター/詳細ビューはメール アプリによく使われる形式で、画面の片側に選択できる項目の一覧を表示し、もう一方の側に選択された項目の詳細ビューを表示します。このビューの例については、次のセクション「適切な対話操作パターンの使用」をご覧ください。

グリッド: グリッド レイアウト (XAML を使ったアプリでは **GridView**) は、常に横方向にパンし、項目は上から下へ、次に左から右へ読むように配置されます。



グリッドは、次のような場合に使います。

- コンテンツ ライブラリを表示する。このビューの例については、「適切な対話操作パターンの使用」をご覧ください。
- アプリで項目をグループ化する必要がある。グループ化は、コレクションの項目が多数ある場合や、項目が自然にグループに分かれる場合 (アルバム別に整理された曲のコレクションなど) にコンテンツを整理するのに役立ちます。
- セマンティック ズームに関連付けられた 2 つのコンテンツ ビューの形式を設定する。

適切な対話操作パターンの使用

一般に、Windows でリストまたはグリッド ビュー コントロールを使う場合の用途は、コンテンツ ライブラリの表示、マスター データと詳細データの表示、ピッカーとしての使用、静的データの表示の 4 種類です。 このいずれかの用途にこれらのコントロールを使う場合、ユーザーはそれが次の対話式操作パターンに従っていることを期待します。

- **コンテンツ ライブラリ パターン**
  コンテンツのコレクション (ライブラリ) を表示する場合に、このパターンを使います。 通常は、画像や動画などのメディアを提示する場合に使います。

コンテンツ ライブラリでの主なユーザー操作は、項目をタップして動作を開始することです。 ほとんどのコンテンツ ライブラリでは、クロススライドと呼ばれる新しいタッチ ジェスチャーを使って項目を選ぶことができます。クロススライドでは、ユーザーがパン方向に対して垂直方向にスワイプして、項目を選択します。たとえば、水平方向にパンするコンテンツ ライブラリでクロススライドを有効にすると、ユーザーが項目を垂直方向にドラッグして選択できるようになります。

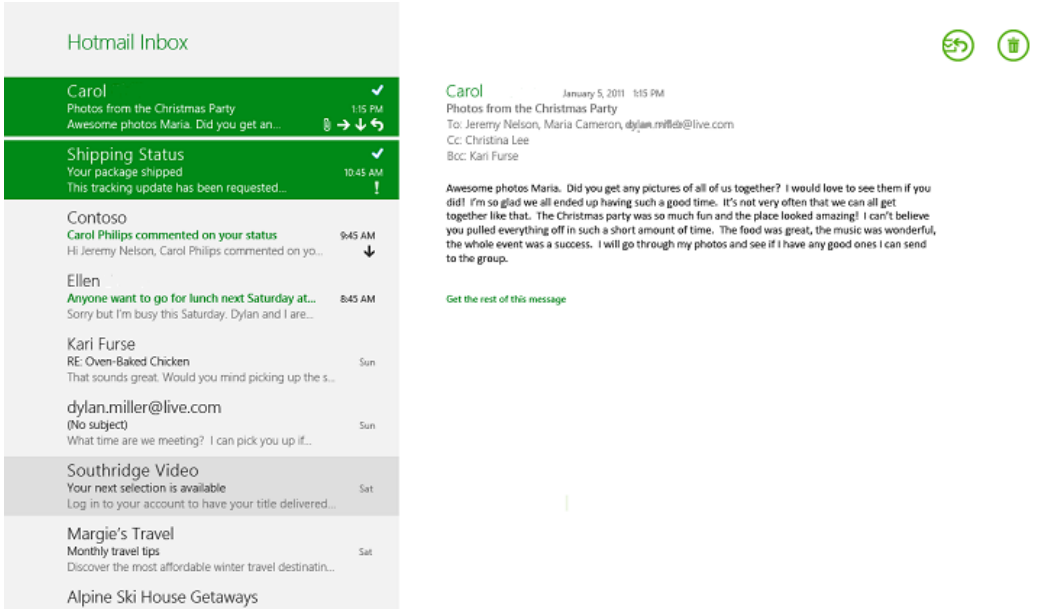
C#/VB/C++ と XAML を使っている場合は、いずれかのグリッド レイアウト用の項目テンプレートを使って形式を設定された **GridView** コントロールを追加することで、このような表示をすばやく実装できます。

JavaScript と HTML を使っている場合は、次のように設定することで、**ListView** にコンテンツ ライブラリ パターンを適用できます。

- **selectionMode:** "single" または "multi" に設定します。
- **tapBehavior:** "invokeOnly" に設定します。
- **swipeBehavior:** 次のいずれかの値に設定します。

- **マスター/詳細パターン (分割ビュー パターンとも呼ばれます)**

  マスター/詳細パターンを使う場合、リスト ビューを使ってマスター ウィンドウを整理できます。マスター ウィンドウには、選択できる項目のリストが表示されます。ユーザーがマスター ウィンドウで項目を選ぶと、それが詳細ページに表示されます。リスト ビューが役に立つのは、マスター ウィンドウの表示形式の設定に対してのみです。詳細ウィンドウに対しては別のコントロールを使います。



XAML の 画像とテキスト リスト (受信トレイ) のテンプレートを使って、マスター ウィンドウ パターンでコンテンツを表示するように **ListView** のスタイルを指定します。

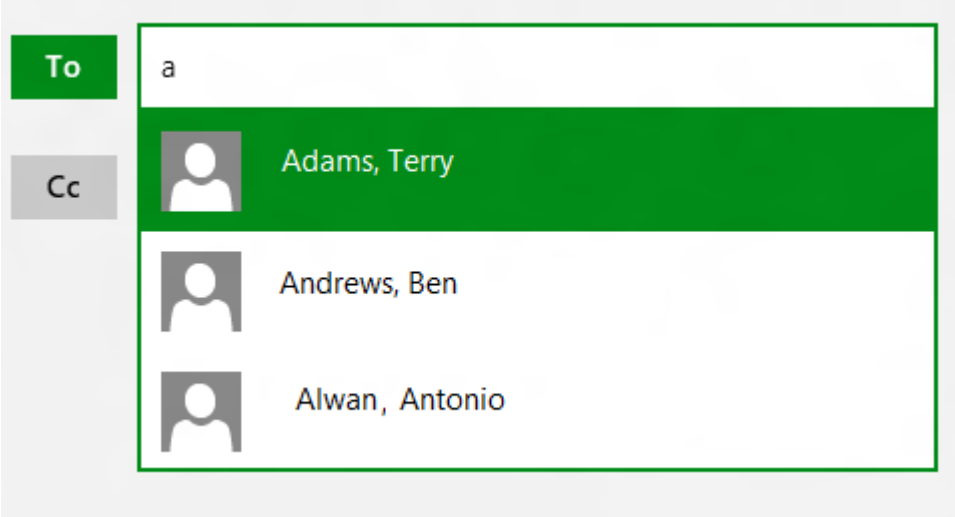
HTML を使う場合、**ListView** でマスター/詳細パターンのマスター ウィンドウのような動作を実現するには、次のように設定します。

- **selectionMode**: "single" または "multi"。単一選択の場合は "single" に設定します。
- **tapBehavior**: "directSelect"
- **swipeBehavior**: 単一選択の場合は "none"、複数選択の場合は "select"。

複数のリスト ビュー コントロールを組み合わせて、マスター/詳細の複雑な階層を作成することができます。

- **ピッカー パターン**

  ピッカー パターンは、ユーザーの主な操作が選択であり、タップで項目を起動する機能が重要ではない場合に使います。この対話操作パターンでは、クロススライドだけでなくタップも選択操作に使われます。



- **静的パターン**
  単にコンテンツを表示する目的のみにリスト ビューを使い、ほとんどの種類の対話操作を使わない場合に、このパターンを使います。 このパターンは、アクティブ化されたりナビゲートされたりしない読み取り専用の項目のコレクションに適しています。

それぞれの対話操作パターンに関連するコードについては、HTML ListView の対話操作のカスタマイズのサンプルに関するページまたは XAML ListView と GridView の対話操作のカスタマイズのサンプに関するページをご覧ください。

推奨と非推奨

- 同じリスト ビュー内の項目は同じ動作を行う必要があります。たとえば、リスト ビュー内の 1 つの項目をユーザーがタップしたときに動作が実行される場合、ユーザーがリスト ビュー内の任意の項目をタップしたときにも動作が実行される必要があります。
  - 一覧がグループに分割されている場合、セマンティック ズームを使います (「**SemanticZoom (HTML)**」または「**SemanticZoom (XAML)**」をご覧ください)。セマンティック ズームを使うと、ユーザーがグループ化されたコンテンツ内を簡単に移動できるようになります。

- 拡大表示ビューを編成するには、WinJS の **ListView** コントロールまたは XAML の **GridView** コントロールでグリッド レイアウトを使います。このビューでは、グループ ヘッダーと個々の項目が表示されます。

- 縮小表示ビューにもグリッド レイアウトを使います。このビューにはグループ ヘッダーのみが表示されます。

- グリッド ビューで対話型のグループ ヘッダーを使う場合、Ctrl + Alt + G のキーの組み合わせを登録し、ユーザーがそれを使って現在フォーカスのある項目のグループに移動できるようにします。
- ユーザーが 1 つの項目しか選択できない場合 (selectionMode が "single" に設定されている場合) は、選択のチェック マークを表示しないでください。 例については、HTML ListView の対話操作のカスタマイズのサンプルに関するページまたは XAML ListView と GridView の対話操作のカスタマイズのサンプに関するページをご覧ください。
- リスト ビューまたはグリッド ビューを汎用のレイアウト コントロールとして使わないでください。グリッド形式や表形式のレイアウトを作成するには、**Hub** コントロール、**Grid** コントロール、またはグリッド レイアウトや可変ボックス

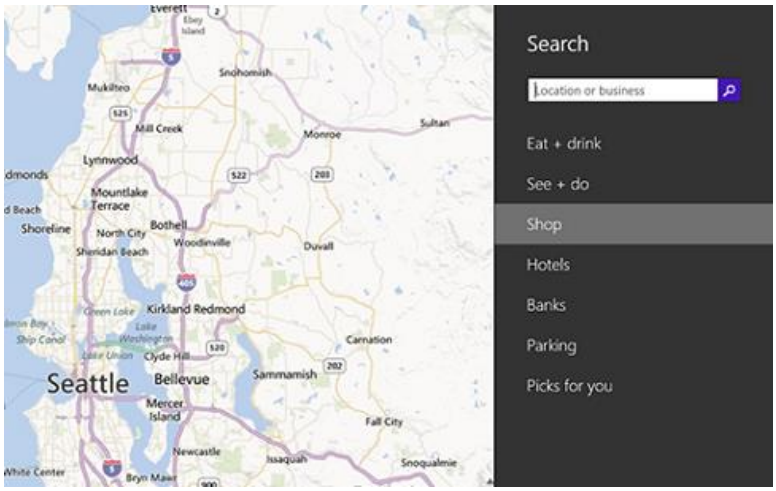
("flexbox") レイアウトなどの CSS レイアウトを使ってください (CSS レイアウトについて詳しくは、「アプリへの CSS3 レイアウトの選択」をご覧ください)。

- 切り取り、コピー、貼り付けの各コマンドに対応するボタンのツール バーなど、コマンド ツール バーを作成するために、リストまたはグリッド ビュー コントロールを使わないでください。 代わりに、ボタン コントロールを並べて配置します。

WinJS の ListView コントロールに固有のガイドライン

- **ListView** でグリッド レイアウトを使う場合、アプリが狭い幅にサイズ変更されたり縦長レイアウトに切り替えられたりしたときは、リスト レイアウトに切り替えます。
- **ListView** にグリッド レイアウトを使いグループが含まれている場合は、リスト レイアウトに切り替えたときにグループを表示します。ユーザーがグループをタップするときに、ListView を更新して、グループ内の項目を表示し、ユーザーがグループ表示に戻るための [戻る] ボタンを表示します。この機能は **SemanticZoom** コントロールを使って実現できます。
- アプリがサイズ変更されるときの **ListView** での項目の表示方法を変更します。アプリがサイズ変更されたビューになったときに、メディア クエリを使って CSS を自動的に更新するか、別の **itemTemplate** に切り替えることができます。CSS スタイルを更新する場合、ListView コントロールに変更が正しく反映されて表示されるように、**forceLayout** メソッドを呼び出します。
- アプリがサイズ変更される前に **ListView** コントロールの **indexOfFirstVisible** 値を格納し、その後、アプリが全画面表示に戻ったときにその値を復元します。これにより、ユーザーが一覧内での位置を見失うことがなくなります。これにより、ユーザーが一覧内での位置を見失うことがなくなります。

マップのガイドライン



Windows アプリ：マップ コントロール

説明

マップ コントロールでは、地図および上空からの写真、方向、検索結果、トラフィックを表示できます。

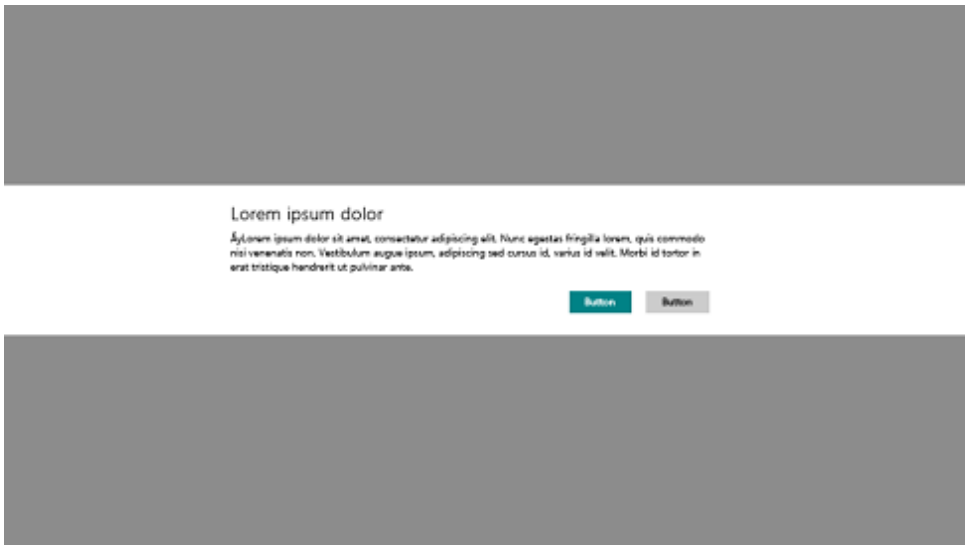
推奨と非推奨

- ユーザーが地理情報を表示するために過度にパンおよびズームを行う必要のない十分な画面領域、理想的には画面領域全体を使います。静的な情報ビューの提示をするためにのみコントロールが使われている場合、より小さなマップを使うことができます。

その他の使い方のガイダンス

ユーザーがアプリを離れなくても、アプリ固有の地理情報または一般的な地理情報を表示できるように、アプリでマップを使います。マップでは、ユーザーの場所、方向、関心のあるスポットを表示できます。またマップでは、上空からの写真、トラフィック、ローカル検索結果を表示できます。

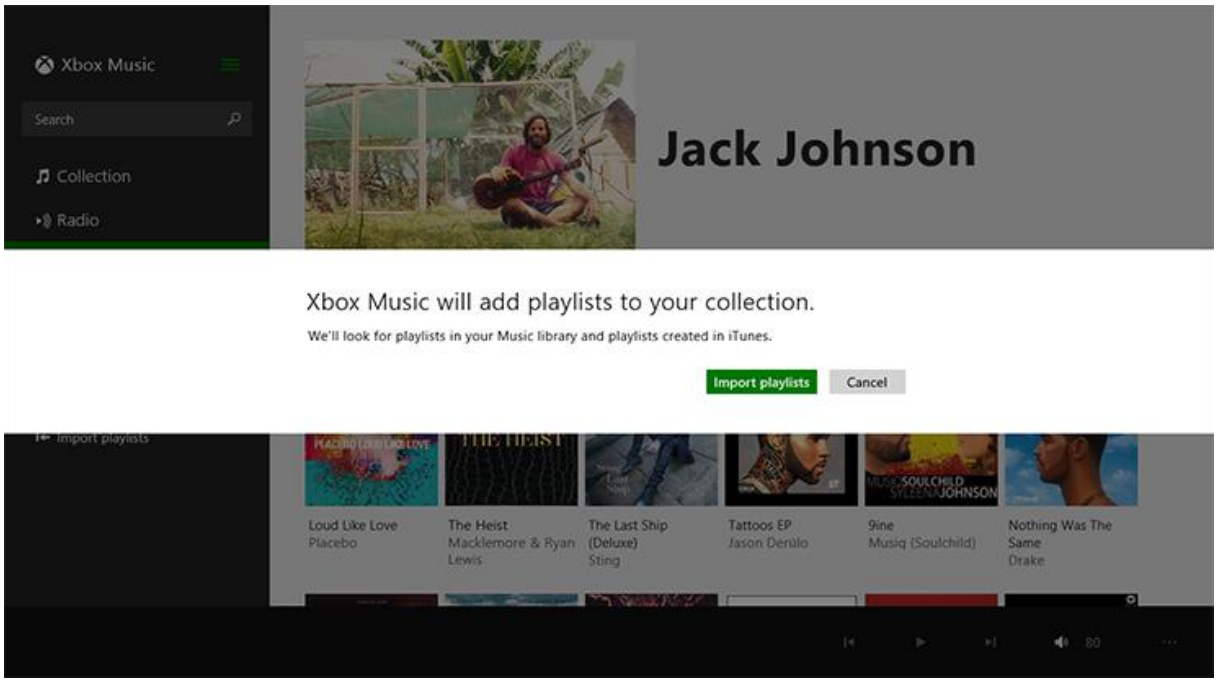
メッセージ ダイアログのガイドライン



説明

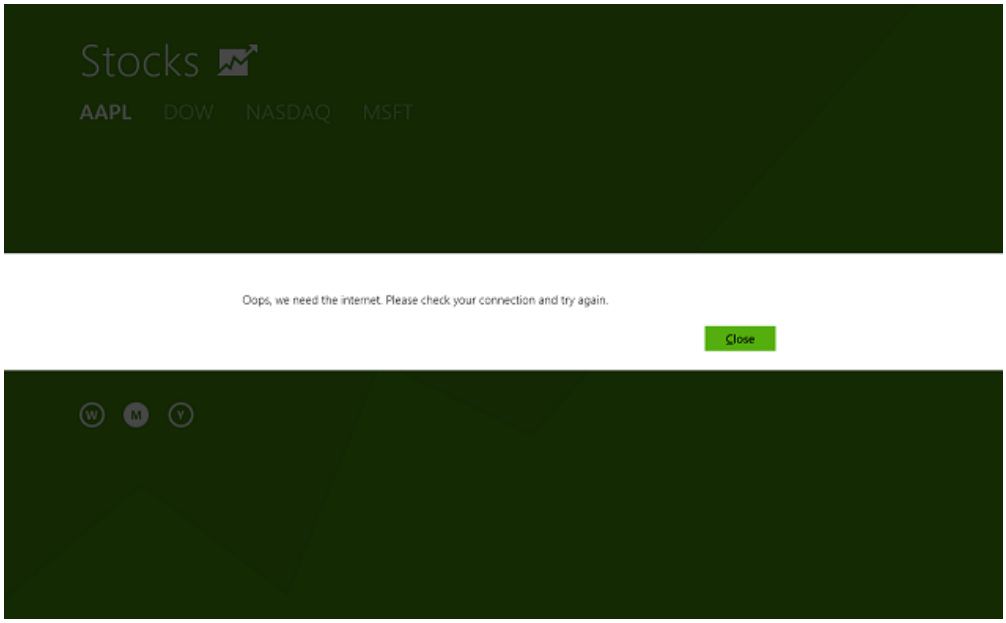
メッセージ ダイアログは、常にモーダルで明示的に閉じられる安定した状況依存のサーフェスを提供する、オーバーレイ UI 要素です。メッセージ ダイアログは、画面上の一貫した場所に表示されます。

例



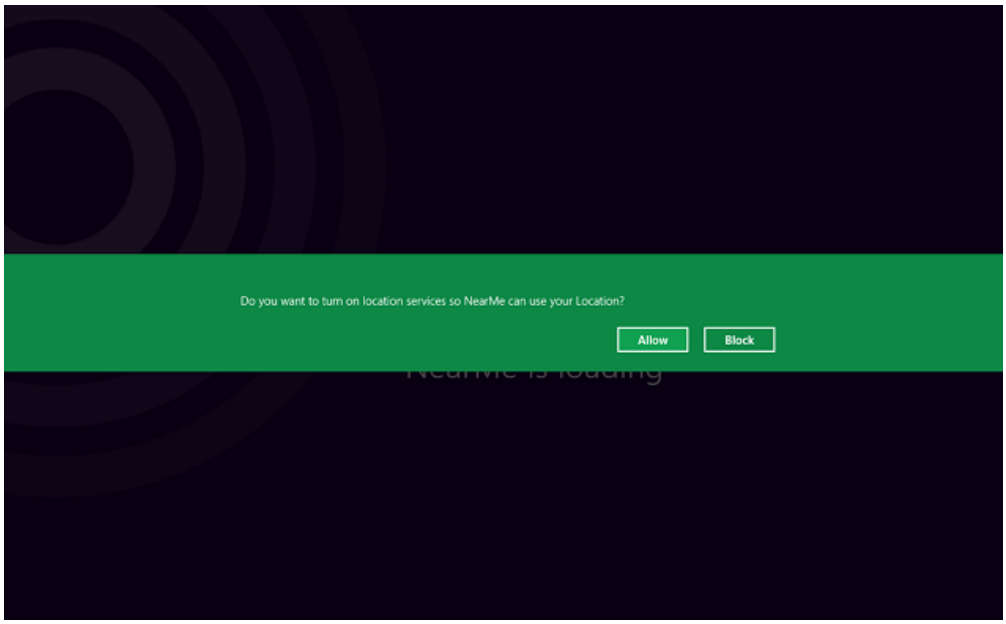
エラー

アプリ全体の状況に適用されるエラー メッセージでは、メッセージ ダイアログを使います。これらは、インラインで伝えられるエラー メッセージとは異なります。適切な例は、接続エラーを表示する、次のようなメッセージ ダイアログです。このエラーは、ユーザーにとってのアプリケーションの価値に重大な影響を与えます。



質問

次の例は、位置情報サービスの使用に対する同意を求める、Windows デバイス同意ブローカーからのメッセージ ダイアログです。



推奨と非推奨

- ユーザーが操作を中断して、見たり認識したりする必要がある緊急の情報を伝えるには、メッセージ ダイアログを使います。たとえば、"高度な機能の試用期間が終了しました" というメッセージです。
- ユーザーの入力を必要とするブロック質問を表示するには、メッセージ ダイアログを使います。ブロック質問とは、アプリケーションがユーザーに代わって選択することができず、回答がないとアプリが動作を継続できなくなるような質問です。ブロック質問では、ユーザーに明確な選択を提示する必要があります。これは、無視したり先送りにしたりできない質問です。
- ユーザーの明示的な操作を求める場合、またはユーザーが認識することが重要であるメッセージを表示する場合は、メッセージ ダイアログを使います。ダイアログの使い方の例は次のとおりです。
    - ユーザーが重要な資産に永続的な変更を加えようとしている場合
    - ユーザーが重要な資産を削除しようとしている場合
    - ユーザーのシステムのセキュリティが侵害される可能性がある場合
- アプリやシステムでその後のアクションに長い時間を費やす必要が生じ、誤って閉じるとユーザーの信頼を損なうような場合は、カスタム ダイアログを使います。
- すべてのダイアログでは、(タイトルの有無に関係なく) ダイアログ内のテキストの 1 行目で、ユーザーの目的 (実行する内容) を明確に示す必要があります。
- ユーザーが行った操作に対するユーザーの意思確認をアプリが必要とする場合、メッセージ ダイアログは使いません。代わりに、フライアウトが適切なサーフェスです。「フライアウトのガイドライン」をご覧ください。
- ページの特定の場所 (たとえばパスワード フィールド) に関連するエラー (検証エラーなど) の場合、メッセージ ダイアログは使わずに、アプリのキャンバス自体を使ってインライン エラーを表示します。「適切な UI サーフェスの選択: エラー」をご覧ください。

その他の使い方のガイダンス

すべてのメッセージ ダイアログでは、ダイアログ内のテキストの 1 行目で、ユーザーの目的 (実行する内容) を明確に示す必要があります。メッセージ ダイアログの "タイトル" フィ

ールドと "コンテンツ" フィールドを使って情報を効率的に伝える方法は、次のガイドラインのとおりです。

- **タイトル (メインの指示、オプション)**
  - 簡潔なタイトルを使って、ユーザーがそのダイアログで行う必要がある操作を説明します。タイトルが長い場合は、折り返されず省略されます。
  - ダイアログを使って、簡単なメッセージ、エラー、または質問を表示する場合は、タイトルを省略することもできます。主な情報はコンテンツのテキストを使って伝えます。
  - タイトルは、ボタンの選択に直接関連するものにします。
- **コンテンツ (説明文)**
  - メッセージ、エラー、または操作をブロック質問をできる限り簡潔に示します。本質的でない情報は表示しません。
  - タイトルを使う場合は、コンテンツ領域を詳しい情報の提示や用語の定義に使います。タイトルの言葉づかいを変えただけの文を繰り返さないようにします。
- **ボタン**
  - テキストを指定したボタンを使って、主な説明またはコンテンツに対する応答を示します。たとえば、主な説明が "使っているコンピューターへの AppName からのアクセスを許可しますか?" の場合、"許可" ボタンと "ブロック" ボタンを使います。具体的な応答の言葉はすばやく理解できるので、効率的に判断できます。
  - "OK" と "キャンセル" のような汎用的なパターンは使わないようにします。
  - ユーザーに最も実行してもらいたい操作を表す既定のボタンを指定します。前に示した例では、"許可" が既定の選択です。
    - 指定しない場合、一番左のボタンが既定のボタンになります。
    - 最も安全で保守的な選択に対応するボタンを一番右に配置します。前に示した例では、"ブロック" が最も保守的な選択であるため、最も右側にあります。
- **色**
  - メッセージ ダイアログの背景は常に白色です。ダイアログを所有しているアプリの基本色は、コントロールに使われます。

ピボット のガイドライン (Windows Phone ストア アプリ)

Windows Phone アプリ：ピボット項目を含むピボット コントロール

説明

ピボット コントロールは、通常は同じデータ セット内の異なるピボット (ビューまたはフィルター) 間で、迅速な移動手段を提供する全画面表示のコンテナーおよびナビゲーション モデルです。たとえば、ピボット コントロールを使った電子メール アプリでは、最初のピボット項目 (またはビュー) 内のすべての電子メールを一覧に示した後、他のピボット項目で同じ一覧を未読の電子メール、フラグ付き電子メール、緊急の電子メールにフィルターできます。

推奨と非推奨

- ピボット コントロールの既定の外観を上書きするためにテーマを使います。
- ピボット コントロールは、最後のピボット項目から最初のピボット項目に、および逆方向にラップします。アプリのフローを正しく設計するためにこの効果を使います。
- パフォーマンス上の理由から、ユーザーが迷う可能性を小さくするために、ピボット コントロール内では 4 ～ 5 個未満のピボット項目のみを使います。ピボット コントロールの使用は控え目にし、エクスペリエンスに適切なシナリオでのみピボット項目を使います。
- 同様の種類のオブジェクトまたはデータを表示する場合にのみピボット コントロールを使います (たとえば、同じデータのフィルター処理されたビューなど)。
- ユーザーに次のピボット ウィンドウの存在についてビジュアルなヒントを提供し、ローカライズに役立つように、ピボット項目のヘッダー テキストは最大で 2 単語に制限します。
- ピボット コントロールをタスク フローには使わないでください (根本的に異なるタスクを示すことになります)。異なるピボット項目はシームレスにフローし (外観)、項目間の移動でユーザー アクティビティを大幅に変更することは避ける必要があります (たとえば、メールをフィルターするページと図を表示するページなど)。
- ユーザー操作によって情報が追加される可能性がある間、空のピボット項目は削除しないでください。たとえば、未読の電子メールが現在ない場合でも、同期によって表示される可能性があるので、未読メールのピボット項目は削除しないでください。その代わりに、"未読メッセージがありません" などのプレースホルダー コンテンツを表示します。
- ハブ コントロール内ではピボット コントロールを使わないでください。逆の場合も同じです。また、ピボット コントロールを別のピボット コントロール内に配置しないでください。ただし、ピボット コントロールに対するハブ セクション リンク内にオブジェクトを含めることができます。逆の場合も同様です。
- ピボット コントロール内でパンやスクロールできるコントロールを使わないでください。たとえば、ピボット項目内にマップ コントロールを配置すると、ピボット コントロールが使いにくくなる場合があります。ジェスチャ入力が混同されま

す。たとえば、ピボット コントロールの項目内にあるスライダーを左にスライドしようとした場合、隣接する項目への移動またはスライダーの移動のいずれを希望しているのかが不明確です。これを解決するには、ジェスチャ入力を必要とするコントロールを個別のページに配置し、そこにナビゲートします。ジェスチャが無効になっているコントロール (おそらくはマップ) はピボット項目に配置しても問題ありません。マップをアクティブ化するボタンをオーバーレイできます。このボタンを押すかタップすると、マップだけが含まれる別のページに移動します。その後、ユーザーは、戻るボタンを押してピボット項目に戻ることができます。

- ピボット項目内でテキスト入力ボックスを使わないでください。この操作は、左から右へのフリックおよびパン ジェスチャ操作に干渉します。

その他の使い方のガイダンス

ピボット コントロールを使って、大きなデータ セットのフィルター処理、複数のデータ セットの表示、アプリ ビューの切り替えを行います。アプリは、統合 Windows Phone ピボット エクスペリエンスのような外観で応答できます。

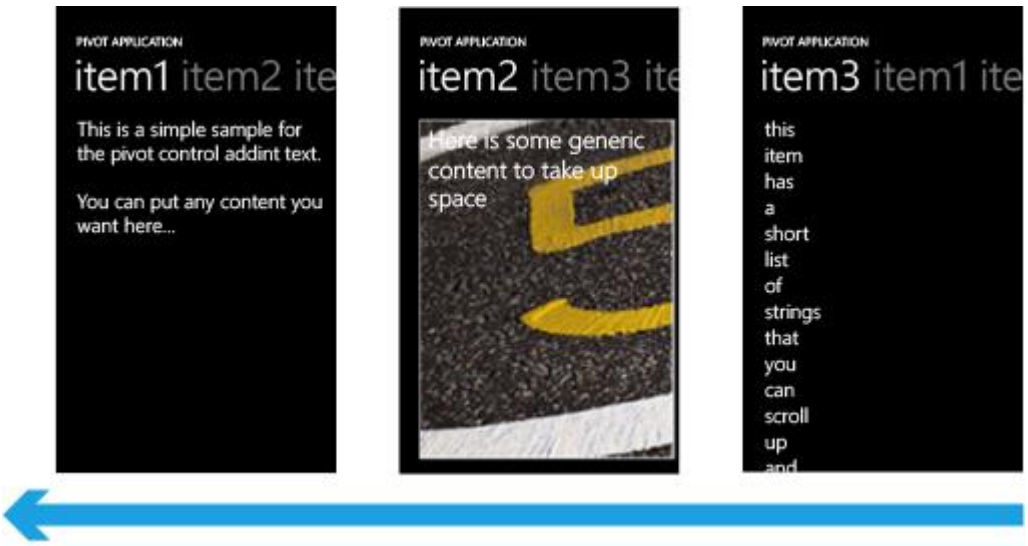
外観と操作

ピボット コントロールは、相互に水平方向に配置された一連のピボット項目 (またはビュー) をホストします。このコントロールによってユーザーは、水平方向にスライドまたはフリックして次のピボット項目または前のピボット項目に進むことができます。

ピボットには、次のタッチ操作のサポートが組み込まれています。

- 水平方向のスライド (タッチおよび左/右のドラッグ)
- 水平方向のフリック (タッチおよび左/右のすばやいフリック)

一般的な場合—たとえばリンクをタップしてリストを垂直方向にスクロールできる場合に、ピボット項目内でホストされたコントロールは引き続き対話的です。



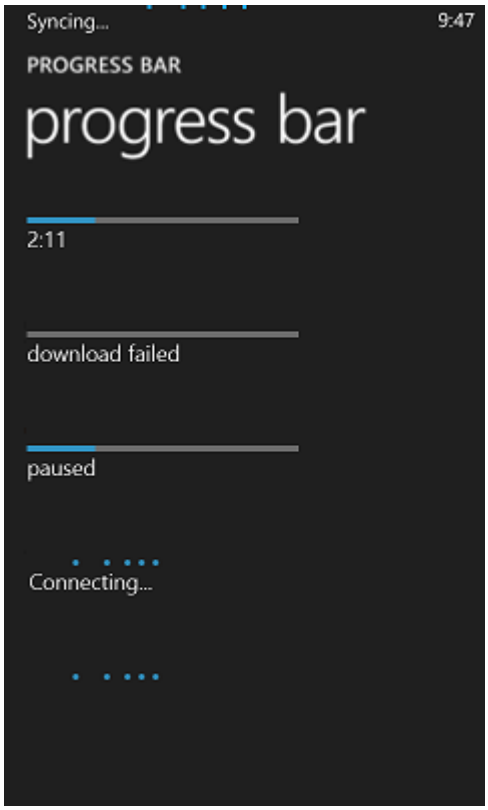
コントロールの説明

ピボット コントロールは、ピボット項目のホスト (つまりコンテナー) です。そして各ピボット項目には、レイアウト パネル、コントロール、リンクなど、任意のコンテンツを含めることができます。ピボット コントロールのアーキテクチャについて詳しくは、「Windows Phone のピボット コントロール アーキテクチャ」をご覧ください。

ピボット コントロールの各部 (ピボット タイトル、ピボット項目ヘッダーなど)

プログレス コントロールのガイドライン

Windows アプリ：進行状況不定バー、進行状況リング、進行状況確定バー



Windows Phone アプリ：ステータス バーの進行状況インジケーターと進行状況バー

説明

プログレス コントロールは、時間のかかる操作が進行中であることを示すフィードバックをユーザーに返します。進行状況確定バーは、操作の完了割合を示します。進行状況不定バーと進行状況リングは、操作が進行中であることを示します。

プログレス コントロールは読み取り専用です。対話型ではありません。

例

スプラッシュ スクリーンの進行状況不定リング コントロールの例を次に示します。

進行状況バーは、状態や位置を示す優れたインジケーターでもあります。音楽トラックで使われる進行状況バーは、曲のタイムラインに対応しています。バーの値は、曲の位置を示します。一時停止状態は、再生が一時停止されていることを示します。

適切なコントロールの選択

常にプログレス コントロールを示す必要があるわけではありません。タスクが進行中であることが十分に明白であったり、タスクがすぐに完了するのでプログレス コントロールを表示するとわずらわしかったりする場合もあります。プログレス コントロールを表示するかどうかを判断するときには、次のような点を考慮します。

- **操作を完了するまでに 2 秒より長くかかるか**
  そうである場合は、操作を開始したらすぐに、プログレス コントロールを表示します。操作の完了までに 2 秒より長い時間が通常かかるとしても、2 秒未満で完了

することもある場合は、ちらつきを避けるために、500 ミリ秒待機してからコントロールを表示します。

- **操作は、ユーザーがタスクを完了するのを待っているか?**
  そうである場合は、プログレス バーを使いません。プログレス バーは、ユーザーの作業ではなく、コンピューターの作業の進行状況を示すものです。
- **何かが行われていることをユーザーが知る必要があるか?**
  たとえば、アプリがバックグラウンドで何かをダウンロードしていて、ダウンロードを開始したのがユーザーでない場合、ユーザーはそのことを知る必要がありません。
- **操作が、ユーザーのアクティビティをブロックしないバックグラウンド アクティビティであり、ユーザーにはほとんど関与しない (少しだけ関与する) か?**
  アプリが、常に見えている必要はないものの、進行状況を表示する必要はあるタスクを実行している場合は、テキストと省略記号を使います。

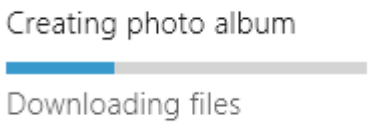
  Sharing in progress...

  タスクが進行中であることを示すために、省略記号を使います。複数のタスクまたは項目がある場合は、残りのタスクの数を示すことができます。すべてのタスクが完了したら、インジケーターを消します。
- **進行状況をビジュアル化するために、操作からのコンテンツを使えるか?**
  使える場合は、プログレス コントロールを表示しません。 たとえば、ディスクから読み込まれる画像を表示する場合は、画像が読み込まれるたびに、画面に 1 つずつ画像が表示されます。プログレス コントロールを表示しても、余分な UI が増えるだけで、何の利点もありません。
- **操作の進行中に、作業が全体に対してどの程度完了したかを決定できるか?**
  その場合は進行状況確定バーを使います。ユーザーをブロックする作業の場合は特に使ってください。それ以外の場合は、進行状況不定バーまたは進行状況リングを使います。ユーザーは何かが行われていることを知るだけですが、それでも十分に有用です。

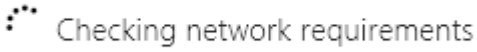
推奨と非推奨

プログレス コントロールには、次の 3 つのスタイルがあります。

- タスクが確定的である場合、つまり、継続時間が明確に定義されていたり、終了が予測可能だったりする場合は、進行状況確定バーを使います。たとえば、時間単位、バイト単位、ファイル単位などの定量化できる測定単位で残りの作業量を推定できる場合は、進行状況確定バーを使います。確定的なタスクには、次のような例があります。
    - アプリが 500 k の写真をダウンロードしていて、これまでに 100 k を受信した。
    - アプリが 15 秒の広告を表示していて、2 秒が経過した。



- ユーザーの操作をブロックするモーダルな確定的でないタスクでは、進行状況不定バーを使います。



- ユーザーの操作をブロックしない非モーダルな確定的でないタスクでは、進行状況不定バーを使います。

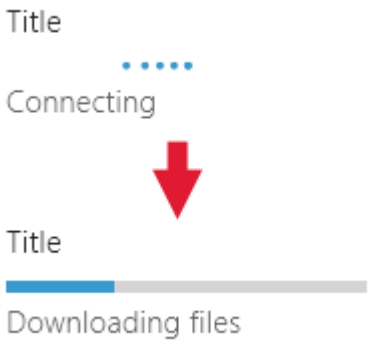
- 一部モーダルなタスクで、モーダルな状態が 2 秒未満である場合は、非モーダルなタスクとして扱います。 一部のタスクは、処理がある程度進むまで操作をブロックし、その後はユーザーがアプリの操作を始められるようになります。 たとえば、ユーザーが検索クエリを実行した場合、最初の結果が表示されるまでは操作がブロックされます。このようなタスクは、モーダルな状態が 2 秒未満である場合は非モーダルとして扱い、進行状況不定バー スタイルを使います。 モーダルな状態が 2 秒より長く続く場合は、タスクのモーダルなフェーズでは進行状況不定リングを使い、非モーダルなフェーズでは進行状況不定バーを使います。

- 進行中の操作をキャンセルするか一時停止するための方法を用意することを検討します。操作が完了するまでユーザーがブロックされるときに、操作の残りの実行時間が明らかな場合は、特に考慮してください。
- アクティビティを示すために、"待機カーソル" は使いません。システムの操作にタッチを使っているユーザーには表示されず、マウスを使っているユーザーにはアクティビティをビジュアル化する方法が 2 つ (カーソルとプログレス コントロール) は必要ないためです。
- 複数のアクティブな関連するタスクには、1 つのプログレス コントロールを表示します。 画面に複数の関連する項目があり、すべてがなんらかのアクティビティを同時に行う場合でも、複数のプログレス コントロールは表示しません。代わりに、最後のタスクの完了時に終了する 1 つのプログレス コントロールを表示します。たとえば、アプリが複数の写真をダウンロードする場合は、写真ごとにプログレス コントロールを表示するのではなく、1 つだけを表示します。
- タスクの実行中は、プログレス コントロールの場所やサイズを変更しません。

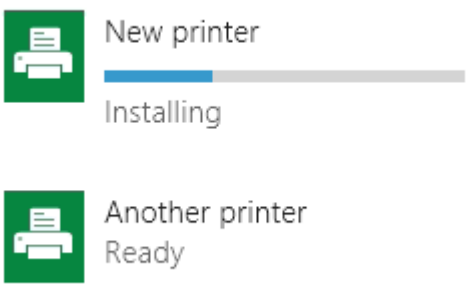
確定的タスクのガイドライン

- 操作がモーダルで (ユーザーの操作がブロックされ)、10 秒より長い時間がかかる場合は、操作を取り消す方法を用意します。操作を取り消すためのオプションは、操作の開始と同時に使えるようにします。
- 進行状況は均等に更新します。 進行状況が 80% を超えた後で長い間停止するような状況にならないようにします。進行が終わりに近づくにつれて、スピードが下がるのではなく、上がることが望まれます。0% から 90% に飛躍するようなことも、ないようにします。
- 進行状況が 100% になった後、進行状況確定バーがアニメーションを終了するまで待機してから、バーを非表示にします。
- タスクはユーザーまたは外部の条件によって停止したが、ユーザーがタスクを再開できる場合は、進行が一時停止されていることをはっきりと示します。JavaScript アプリでは、win-paused CSS スタイルを使います。C#/C++/VB アプリでは、ShowPaused プロパティを true に設定します。進行状況バーの下に、何が起こっているかを示す状態テキストを示します。

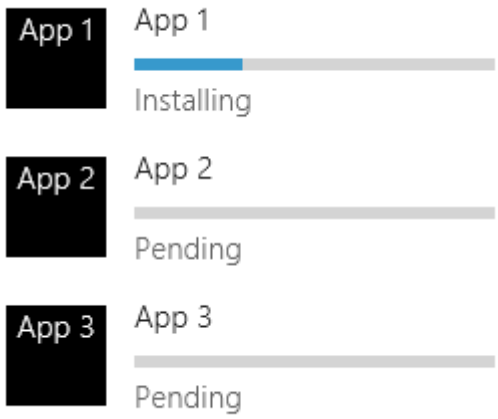
- タスクが停止し、再開できないか始めからやり直す必要がある場合は、エラーが発生したことをはっきりと示します。 JavaScript アプリでは、win-error CSS スタイルを使います。C#/C++/VB アプリでは、ShowError プロパティを true に設定します。バーの下の状態テキストを、何が起きてどのように問題に対処すればよいか (可能な場合) をユーザーに知らせるメッセージに置き換えます。
- 進行状況確定バーを表示する前に、多少の時間 (またはなんらかのアクション) が必要な場合は、まず進行状況不定バーを使い、その後で進行状況確定バーに切り替えます。たとえば、ダウンロード タスクの最初の手順がサーバーへの接続である場合、接続にかかる時間は推定できません。接続の確立後に、進行状況確定バーに切り替えて、ダウンロードの進行状況を表示します。切り替え後も、進行状況バーは正確に同じ位置、同じサイズになるようにします。



- プリンター一覧のような項目の一覧があるときに、一覧の項目に対してなんらかの操作 (いずれかのプリンター用のドライバーのインストールなど) を開始できる場合は、項目の横に進行状況確定バーを表示します。
  進行状況バーの上にタスクの表題 (ラベル)、下に状態を表示します。何が起きているかが明白な場合は、状態テキストは表示しません。 タスクの完了後は、進行状況バーを非表示にします。状態テキストは、項目の新しい状態を知らせるために使います。



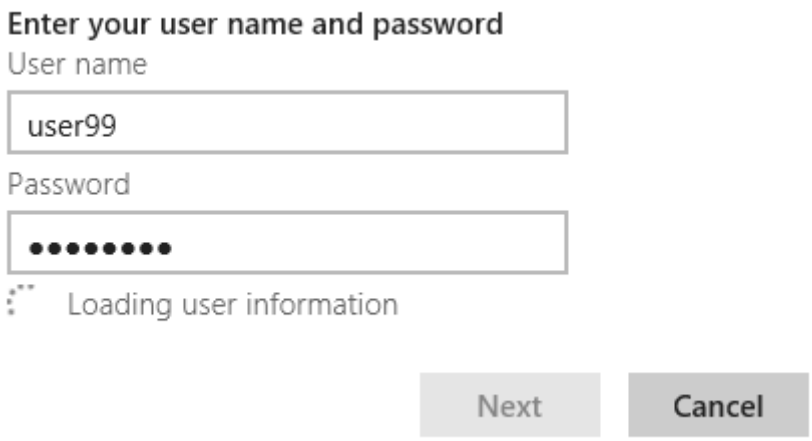
- タスク一覧を表示するには、コンテンツをグリッド内で整列させ、ユーザーが状態をひとめで見られるようにします。保留中の項目も含めて、すべての項目の進行状況バーを表示します。
  この一覧の目的は進行中の操作を示すことなので、完了した操作は一覧から削除します。



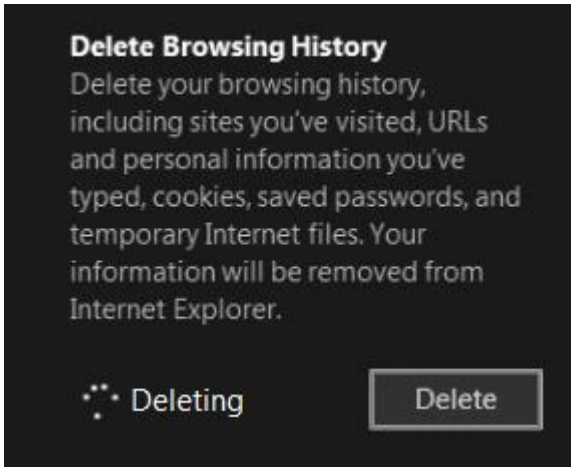
- ユーザーがアプリ バーからタスクを開始し、ユーザーの操作がブロックされた場合は、アプリ バーにプログレス コントロールを表示します。
  進行状況バーが何の進行状況を示しているかが明白な場合は、進行状況バーをアプリ バーの上部に配置して、ラベルと状態は省略できます。そうでない場合は、ラベルと状態テキストを表示します。
  アプリ バーのコントロールを無効にし、コンテンツ領域への入力を無視して、タスクの間は操作を無効にします。
- 進行状況を後退させないでください。 進行状況の値は常に増やします。 操作を元に戻さなければならない場合は、他の操作の進行状況を示すのと同じように、元に戻す処理の進行状況を示します。
- 現在の手順またはタスクが最後のものでないことがユーザーに明白でない場合は、進行状況を再開始 (100% から 0% に) しないようにします。たとえば、データのダウンロードと、そのデータの処理および表示という 2 つの部分がタスクにあるとします。ダウンロードが完了した後、進行状況バーを 0% にリセットし、データ処理の進行状況の表示を始めます。 タスクに複数の手順が含まれていることがユーザーに明白でない場合は、タスクを 1 つの 0 ～ 100% の尺度にまとめて、1 つのタスクから次のタスクに移る際に状態テキストを更新します。

進行状況リングを使う不定的なモーダル タスクのガイドライン

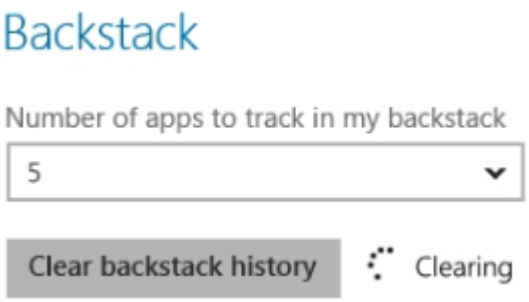
- 操作のコンテキスト内に進行状況リングを表示します。つまり、ユーザーが操作を開始した場所または結果のデータが表示される場所の近くにリングを表示します。
- 進行状況リングの右側に状態テキストを示します。
- 進行状況リングの色を状態テキストの色と同じにします。
- タスクの実行中にユーザーが操作してはいけないコントロールを無効にします。
- タスクの結果がエラーになった場合は、進行状況インジケーターと状態テキストを非表示にして、その場所にエラー メッセージを表示します。
- ダイアログでは、次の画面に移動する前に操作を完了する必要がある場合は、進行状況リングをボタン領域のすぐ上に、ダイアログのコンテンツと左揃えで配置します。



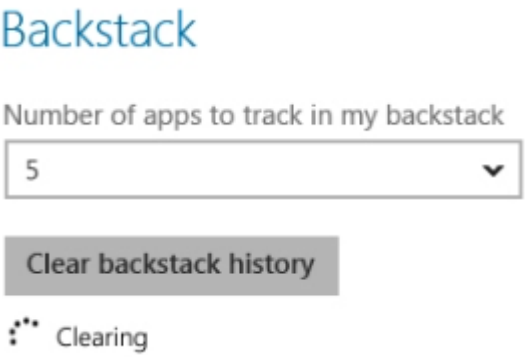
- 右揃えのコントロールがあるアプリ ウィンドウでは、進行状況リングを、処理を引き起こしたコントロールの左またはすぐ上に配置します。進行状況リングを、関連するコンテンツと左揃えで配置します。



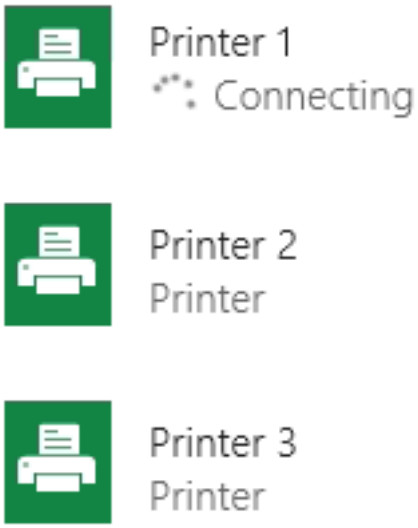
- 左揃えのコントロールがあるアプリ ウィンドウでは、進行状況リングを、処理を引き起こしたコントロールの右またはすぐ下に配置します。



- または –

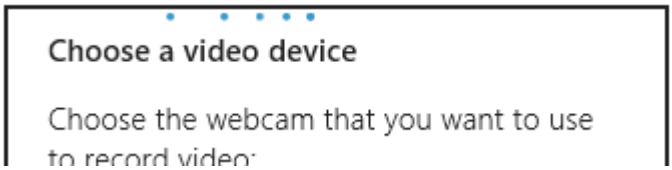


- 複数の項目を表示している場合は、項目のタイトルの下に、進行状況リングと状態テキストを配置します。エラーが発生した場合は、進行状況リングと状態をエラーテキストに置き換えます。

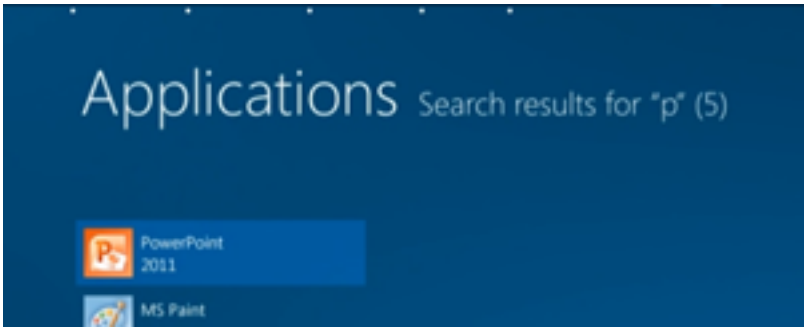


進行状況リングを使う不定的な非モーダル タスクのガイドライン

- フライアウトの中に進行状況を表示する場合は、フライアウトの上部に進行状況不定バーを配置し、幅はフライアウト全体にわたるように設定します。 このように配置すると、不必要に目立つことなく、進行中のアクティビティを示すことができます。 フライアウトにタイトルを表示すると、上部に進行状況バーを配置できなくなるので、タイトルは表示しません。



- アプリ ウィンドウに進行状況を表示する場合は、アプリ ウィンドウの最上部に、ウィンドウ全体にわたるように進行状況不定バーを配置します。



状態テキストのガイドライン

- 進行状況確定バーを使う場合は、状態テキストに進行状況の割合を表示しません。コントロール自体に、その情報が含まれています。
- プログレス コントロールなしでアクティビティを示すためにテキストを使う場合は、アクティビティが進行中であることを表すために省略記号を使います。
- プログレス コントロールを使う場合は、操作が進行中であることをプログレス コントロール自体が示しているので、状態テキストでは省略記号を使いません。

外観とレイアウトのガイドライン

- 進行状況確定バーは、背景が灰色のバーの中を徐々に埋めていく色付きのバーとして表示されます。色付きのバーの全長部分が、操作がどの程度完了したかを相対的に示します。

- 進行状況不定バーまたは進行状況リングは、常に移動する色付きの点で構成されます。
- プログレス コントロールをどの程度目立つ位置に配置するかどうかは、その重要性に基づいて決定します。

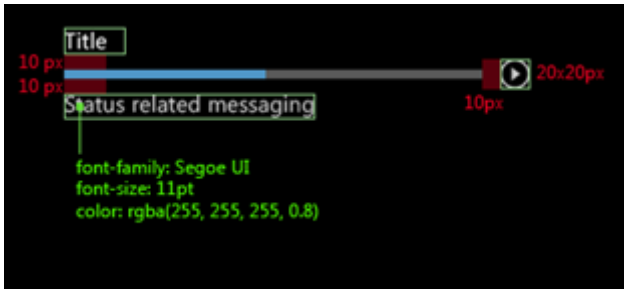
  重要なプログレス コントロールは特定の行動を促す印として使うことができ、システムが作業を完了した後で特定の操作が再開されることをユーザーに伝えます。

  組み込みの Windows Phone アプリの一部では、重要度に応じて、画面の上部にステータス バーの進行状況インジケーターを使っています。同じようにすることができ、確定または不定になるように構成できます。

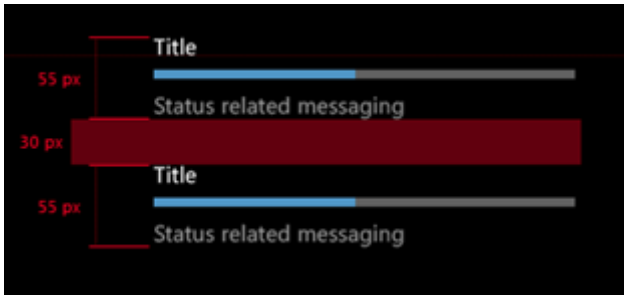
  重要度が低い場合 (ダウンロード中など) は、小さいサイズのコントロールを 1 つのビューに限定して表示します。
- ラベルを使って、進行状況の値を表示するか、進行中の処理を説明するか、処理が中断されていることを示します。ラベルはオプションですが、使うことを強くお勧めします。

  進行中の処理を説明する場合は、動名詞を使います (例: '接続中'、'ダウンロード中'、'送信中')。

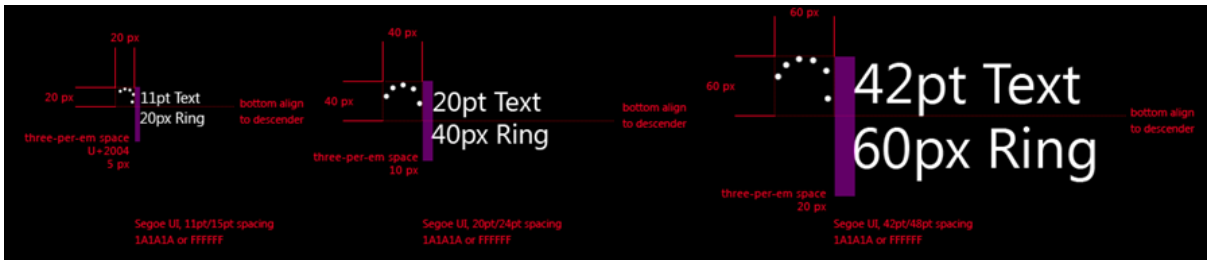
  進行が一時停止したり例外が発生しりした場合は、過去形を使います (例: '一時停止しました'、'ダウンロードは失敗しました'、'取り消されました')。
- ラベルと状態付きの進行状況確定バー



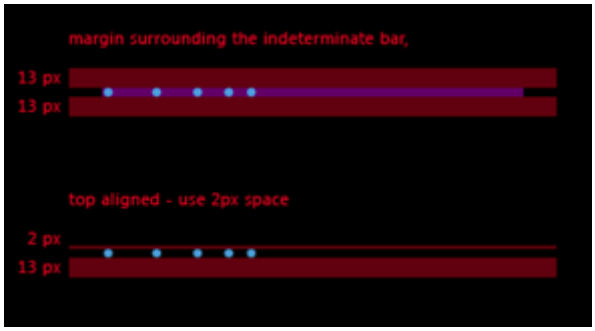
- 複数の進行状況バー



- 状態テキスト付きの進行状況不定リング



- 進行状況不定バー

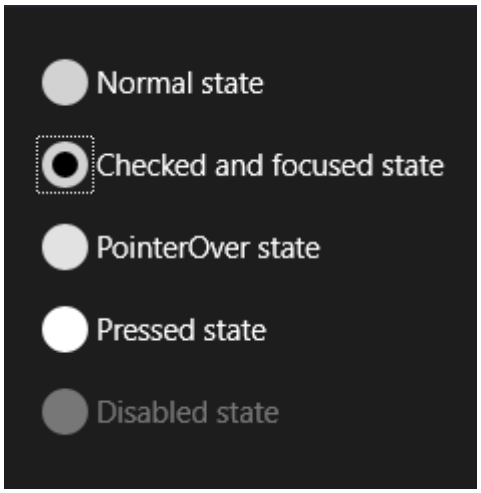


その他の使い方のガイダンス

進行状況のスタイルを選ぶためのデシジョン ツリー

- **何かが行われていることをユーザーが知る必要があるか?**
  必要がない場合は、プログレス コントロールを表示しません。
- **タスクの完了に要する時間に関する情報があるか?**
  - **ある場合:** タスクを完了するまでに 2 秒より長くかかるか?
    - **かかる場合:** 進行状況確定バーを使います。10 秒より長くかかるタスクの場合は、タスクを取り消す方法を用意します。
    - **かからない場合:** プログレス コントロールを表示しません。
  - **ない場合:** タスクが完了するまでユーザーは UI を操作できないか?
    - **できない場合:** このタスクは、操作の特定の詳細をユーザーが認識する必要がある複数手順の一部か?
      - **そうである場合:** 画面の中央に水平に配置された状態テキストがある進行状況不定リングを使います。
      - **そうではない場合:** 画面の中央にテキストのない進行状況不定リングを使います。
    - **できる場合:** これは主要アクティビティか?
      - **そうである場合:** 進行状況は UI の特定の 1 つの要素に関連しているか?
        - **関連している場合:** その関連する UI 要素の横に状態テキストがあるインラインの進行状況不定リングを使います。
        - **関連していない場合:** 大量のデータが一覧に読み込まれているか?
          - **読み込まれている場合:** 受信したコンテンツを表すプレースホルダーの上部の進行状況不定バーを使います。
          - **読み込まれていない場合:** 画面またはサーフェイスの上部の進行状況不定バーを使います。
      - **そうではない場合:** 画面の上隅の状態テキストを使います。

ラジオ ボタンのガイドライン



Windows アプリ：ラジオ ボタン

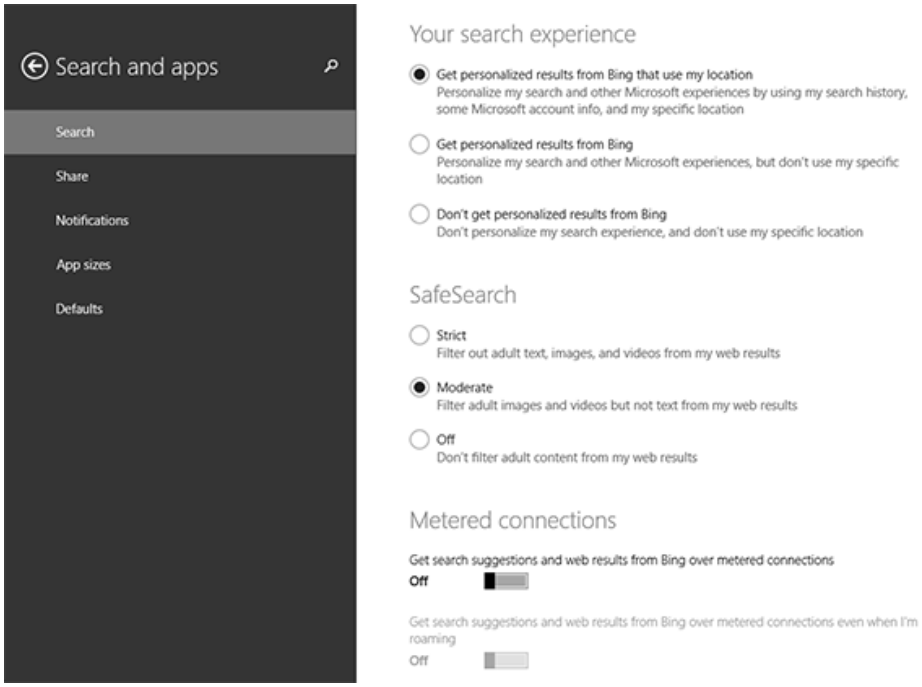
Windows Phone アプリ：ラジオ ボタン

説明

ラジオ ボタンでは、ユーザーは 2 つ以上の選択肢から 1 つのオプションを選ぶことができます。各オプションは、1 つのラジオ ボタンによって表されます。ユーザーは、ラジオ ボタン グループの中から、1 つのラジオ ボタンだけを選ぶことができます。

ラジオ ボタンという名称は、ラジオのチャンネル プリセットのボタンから付けられました。

例



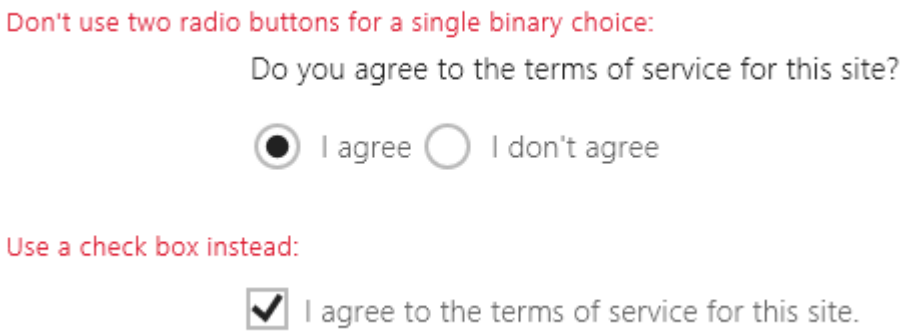
適切なコントロールの選択

ユーザーに 2 つ以上の相互排他的なオプションを提示するには、次のようにラジオ ボタンを使います。

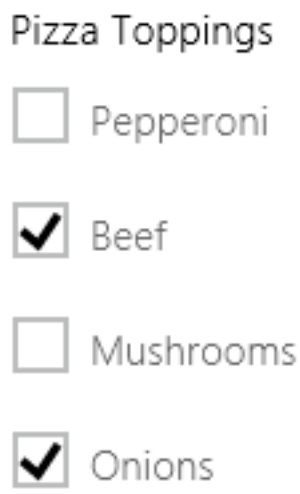
ラジオ ボタンはわかりやすく、アプリで重要なオプションを目立つように表示します。ラジオ ボタンは、広い画面領域を使うに値する重要なオプションを表示する場合であって、選択肢が明確なためにわかりやすいオプション表示が必要な場合に使用してます。

ラジオ ボタンはすべてのオプションを均等に強調するため、必要以上に注目される可能性があります。ユーザーの特別な注目を引く必要がない場合は、他のコントロールを使うことを検討してください。たとえば、 ほとんどの状況でほとんどのユーザーに既定のオプションが適切な場合は、代わりにドロップダウン リストを使います。

2 つだけの相互排他的なオプションの場合は、1 つのチェック ボックスまたはトグル スイッチにまとめます。たとえば、"I agree" と "I don't agree" という 2 つのラジオ ボタンではなく、"I agree" のチェック ボックスを使います。



ユーザーが複数のオプションを選択できる場合は、代わりにチェック ボックスまたはリスト ボックス コントロールを使います。



オプションが 10、20、30 のように固定間隔の数値である場合は、ラジオ ボタンを使いません。代わりに、スライダー コントロールを使います。

オプションが 8 個より多い場合は、ドロップダウン リスト、単一選択のリスト ボックス、または一覧ビューを使います。

オプションがアプリの現在のコンテキストに基づいて表示される場合や、その他の方法で動的に変化する場合は、単一選択のリスト ボックスを使います。

Windows: キーボード経由でアクセスした場合、ラジオ ボタンのグループは、1 つのコントロールのように動作します。Tab キーを使うと選んだオプションにのみアクセスできますが、方向キーを使ってグループを切り替えることができます。

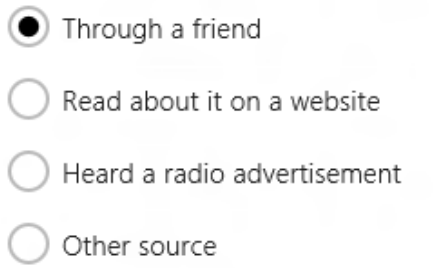
推奨と非推奨

- 一連のラジオ ボタンの用途と現在の状態が明確に表示されていることを確認します。
- ユーザーがラジオ ボタンをタップしたときには、必ずビジュアルなフィードバックを返します。
- ユーザーのラジオ ボタンの操作に合わせて、ビジュアルなフィードバックを返します。ラジオ ボタンの状態には、たとえば、標準、押された状態、オンの状態、オフの状態があります。 ユーザーは、ラジオ ボタンをタップして関連のオプションをアクティブ化します。アクティブなオプションをもう一度タップしても非アクティブにはなりませんが、別のオプションをタップすると、そのオプションにアクティブ化の状態が移ります。
- タッチに対するフィードバック用とオンの状態用にビジュアル効果やアニメーションを予約します。オフの状態のラジオ ボタン コントロールは、使われていない、または非アクティブなコントロールとして表示します (無効なコントロールとして表示しないでください)。
- ラジオ ボタンのテキスト コンテンツは、1 行に収まるように作成します。ラジオ ボタンのビジュアル効果をカスタマイズして、メインのテキスト行の下に小さいフォント サイズでオプションの説明を表示することができます。
- テキスト コンテンツが動的な場合、ボタンのサイズがどのように変わり、周囲のビジュアル効果にどのような影響が生じるかを検討してください。
- ブランドのガイドラインで別のフォントが指示されていない限り、既定のフォントを使います。
- ラベルをタップするとラジオ ボタンが選択されるように、ラベル要素でラジオ ボタンを囲みます。
- ラベル テキストは、ラジオ ボタン コントロールの前や上ではなく、後に配置します。

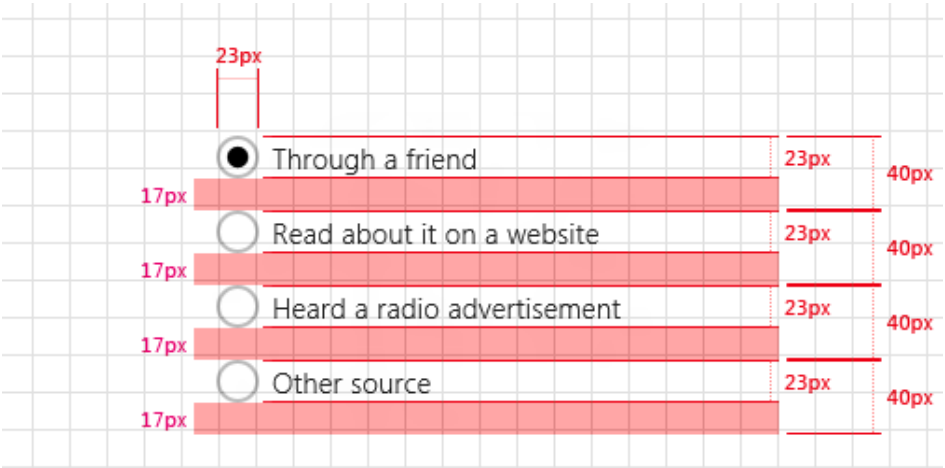
- ラジオ ボタンをカスタマイズすることを検討してください。既定のラジオ ボタンは、2 つの同心円 (内側の円は塗りつぶされ、オンの場合に表示。外側の円はストロークのみ) とテキスト コンテンツで構成されています。しかし、工夫しだいで使いやすさが向上します。アプリのコンテンツを直接操作できれば、ユーザーが理解しやすくなります。たとえば、グラフィックやさりげないテキストのトグル ボタンを使って、提供する実際のコンテンツを表示することもできます。
- ラジオ ボタン グループには、8 個以上のオプションを含めないでください。それより多くのオプションを提示する必要がある場合は、代わりにドロップダウン リスト、リスト ボックス、リスト ビューなどを使います。
- 2 つのラジオ ボタン グループを並べて配置しないようにします。 2 つのラジオ ボタン グループが並んでいると、どのボタンがどのグループに属しているかがわかりにくくなります。グループを分けるには、グループ ラベルを使います。

その他の使い方のガイダンス

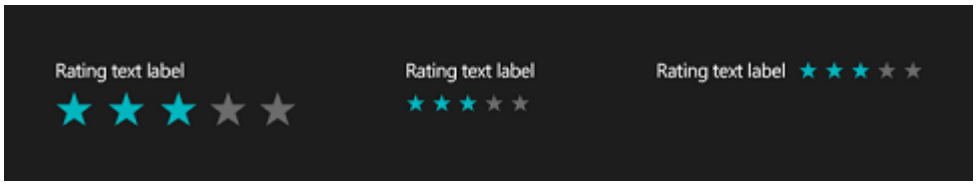
この図は、適切な位置と間隔で配置したラジオ ボタンを示しています。



この図は、前の図で使ったサイズや間隔を具体的に示しています。



評価コントロールのガイドライン

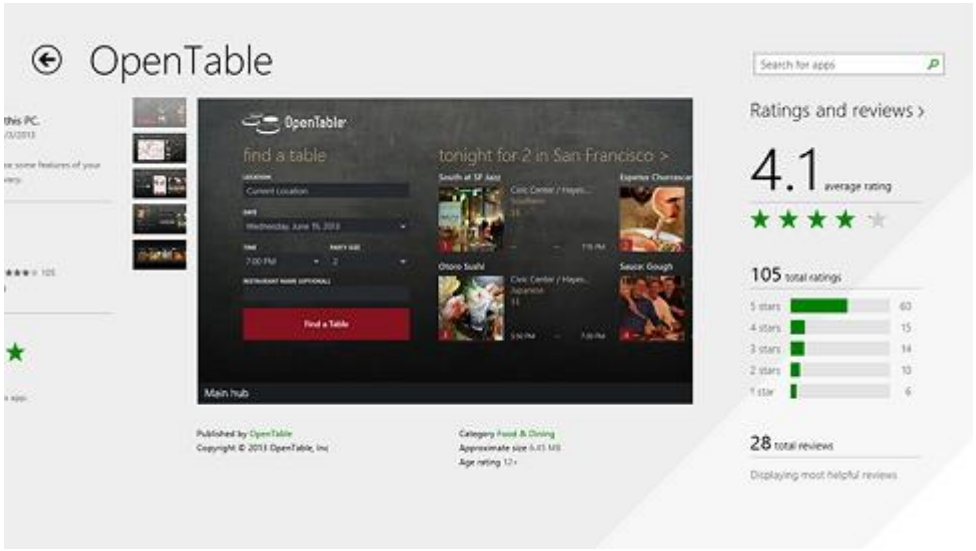


説明

評価コントロールは、評価を示すアイコンをクリックすることで、ユーザーが何かを評価できるようにします。 評価には、平均評価、暫定評価、ユーザー評価の 3 種類があります。

注 評価コントロールは HTML でのみ利用できます。XAML の評価コントロールはありません。

例



適切なコントロールの選択

項目またはサービスに対するユーザーの好感度や満足度を表示するには、評価コントロールを使います。たとえば、評価コントロールを使って、ユーザーが映画を評価できるようにします。 画面の明るさなどのように、連続的な範囲を持つ他の種類のデータには使いません (代わりにスライダーを使います)。

評価コントロールをフィルター コントロールとしては使いません。たとえば、ユーザーが検索結果をフィルター処理して、5 つ星のレストランを表示できるようにする場合に、評価コントロールをフィルターとして実装しないようにします。評価コントロールを使うと、ユーザーがレストランに新しい評価を設定しているのだと誤解する可能性があります。このような場合、代わりにドロップダウン リストを使用できます。

1 つ星評価コントロールを、好き/嫌いを表すコントロールとして使いません。代わりに、チェック ボックスを使います。評価コントロールは、二肢選択評価用には設計されていません。たとえば、コントロールをタップしても、星のオンとオフを切り替えることはできません。JavaScript を使った Windows ストア アプリでの好き/嫌いを表すコントロールとしてチェック ボックスを使う方法の例については、 一般的な HTML コントロールと日常的なウィジェットの例のページをご覧ください。

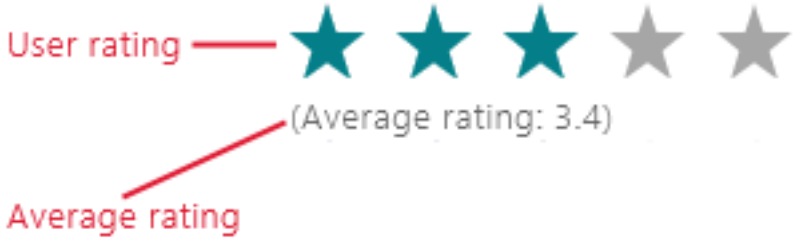
推奨と非推奨

- わかりやすくするための情報をユーザーに提供するには、ツールチップを使います。ツールチップをカスタマイズして、次の図のように、"すばらしい"、"とても良い"、"まあまあ" など、各星の意味を表示できます。

- ユーザーが評価を追加または変更しないようにする場合は、評価コントロールを無効化します。 無効化された評価コントロールには引き続き評価が表示されますが (設定されている場合)、ユーザーは評価の追加や変更ができなくなります。たとえば、評価をログイン ユーザーだけに限定するとします。その場合は、評価コントロールを無効化して、ユーザーがコントロールをタップしたら、ログイン ページが表示されるようにします。

  コントロールを有効化できない場合 (アプリで常に読み取り専用である場合) は、コントロールのホスト要素の class 属性を "win-small" に設定して、他の評価コントロールよりも小さくします。 コントロールを小さくすると、他のコントロールと区別しやすくなり、操作対象でないこともわかりやすくなります。

- 平均評価とユーザー評価を同時に表示します。 ユーザーが評価を指定すると、評価コントロールは平均評価ではなく、ユーザーの評価を表示します。ユーザーにとって意味がある場合は常に、ユーザーの評価の他に平均ユーザー評価も表示します。平均評価を表示するには、次の 2 つの方法があります。
  - 付随するテキスト文字列に平均を表示します ("平均: 3.5" など)。
  - 2 つの評価コントロールを一緒に使い、1 つにはユーザー評価を表示します。もう 1 つには平均評価を表示し、ユーザーの入力は許可しません。

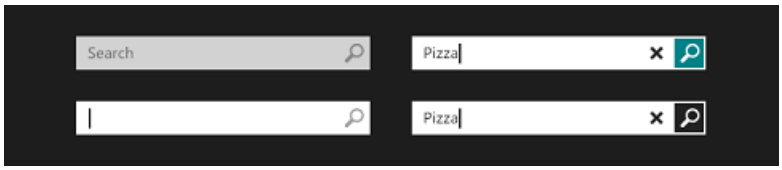


- 必要でない限り、既定の星の数 (最高評価) を変更しません。既定では、評価コントロールには 5 つの星があります。1 が最低、最悪の評価で、5 が最高、最良の評価

です。 アプリがこの慣例に従っていると、ユーザーは評価の意味を簡単に理解できます。

- ユーザーが自分の評価を削除できないようにする必要がない限り、"自分の評価のクリア" 機能を無効にしません。

検索のガイドライン



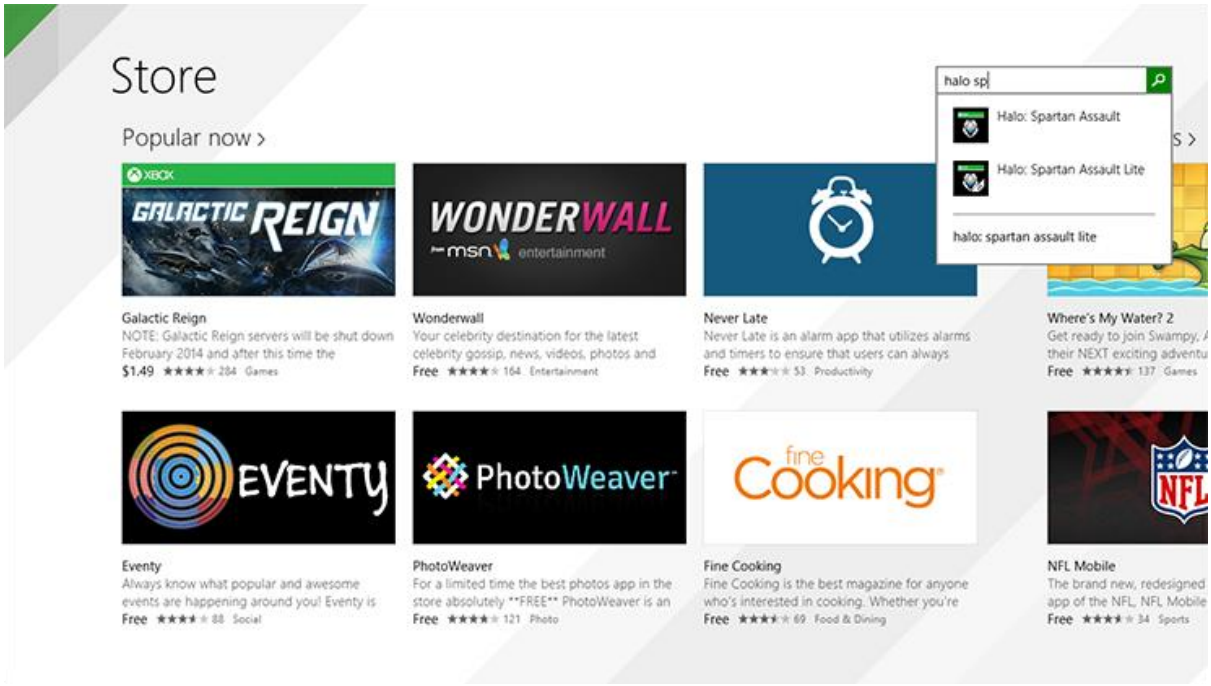
アプリのコンテンツに検索結果を提供するには、次のガイドラインに従ってください。このガイドラインには、検索ボックスで検索候補とプレースホルダー テキストを提供する場合や検索結果ページを設計する場合のヘルプを記載しています。

説明

検索ボックスは、検索クエリ テキストを入力するのに使うことのできるコントロールです。

アプリでコンテンツを参照する主な方法の 1 つが検索である場合、アプリのキャンバスに検索ボックスを追加することを検討してください。検索ボックスは、作業を開始する場所をユーザーに伝える方法としても優れています。レイアウトでスペースの問題がある場合は、展開すると検索ボックスを表示するアイコンを使います。また、アプリ バーに検索ボックスを配置することもできます。アプリの各ページに検索ボックスを配置することを検討します。一部のページには検索ボックスがあり、他のページにはない場合、ユーザーが混乱する可能性があります。ただし、この決定は、コンテキストとアプリの目的によって異なります。

例



検索の実装

アプリにユーザーが検索するコンテンツがある場合は、アプリのキャンバスに検索ボックスを追加します。

- C++、C#、または Visual Basic を使った Windows ストア アプリでは **SearchBox** コントロールを使います。JavaScript を使った Windows ストア アプリでは **WinJS.UI.SearchBox** オブジェクトを使います。
- アプリのコンテンツの配置場所 (ローカル ファイル システムや Web サービスなど) に関係なく、検索チャームを使ってユーザーのクエリに応答し、独自のデザインのアプリ ページに検索結果を表示できます。
- 検索ボックスは、作業を開始する場所をユーザーに伝える方法としても優れています。レイアウトでスペースの問題がある場合は、展開すると検索ボックスを表示するアイコンを使います。また、アプリ バーに検索ボックスを配置することもできます。
- アプリ バーで検索 UI が必要な場合は、上部のアプリ バーに配置します。JavaScript を使う Windows ストア アプリでは、ナビゲーション バーに検索 UI を配置できます。

- アプリの各ページに検索ボックスを配置することを検討します。一部のページには検索ボックスがあり、他のページにはない場合、ユーザーが混乱する可能性があります。ただし、この決定は、コンテキストとアプリの目的によって異なります。

検索ボックスの候補とプレースホルダー テキストのカスタマイズ

ユーザーが検索ボックスにクエリを入力し始めたら、検索ボックスの下のフライアウトに検索候補を表示します。アプリでこれらの機能を提供するには、**AppendResultSuggestion** メソッドと **AppendQuerySuggestion** メソッドを使います。

検索ボックスは最大 25 個の検索候補を受け取ります。

アプリでは、クエリ候補と結果候補の 2 種類の検索候補を提供できます。結果候補は回答、クエリ候補は可能性と考えることができます。

- **クエリ候補**は、ユーザーのクエリ テキストがオート コンプリートされたテキストであり、ユーザーが検索しようとしているクエリを予想して示します。たとえば、ユーザーがストア アプリでアプリを検索していて、ストア アプリの検索ボックスに「word」と入力したとします。"Wordament" や "Wordpress" が結果候補として、"word games" や "words" がクエリ候補として表示されます。 ユーザーがいずれかのクエリ候補を選ぶと、ストア アプリではそのクエリの検索結果ページがユーザーに表示されます。通常、クエリ候補はオート コンプリートされますが、オート コンプリートは必須ではありません。たとえば、レシピ アプリで "eggs (玉子)" を検索する場合、"quiche (キッシュ)" は有効なクエリ候補です。
- **結果候補**は、ユーザーのクエリに大部分が一致するか完全に一致する候補で、即座に表示されます。たとえば、"word" をクエリ対象とした結果として、Recommendations ラベルの下に Wordament アプリが示されます。ユーザーが結果候補の Wordament を選ぶと、ストア アプリではストア内の Wordament アプリのページがユーザーに直接表示されます。

結果候補は、通常、より近い候補であるため、候補のフライアウトの先頭に表示され、その後にクエリ候補が続きます。

ユーザーが検索クエリを入力すると、システムによって、検索履歴にある予想されるクエリの候補が検索ウィンドウに自動的に表示されます。これらの候補は、そのアプリの検索ボッ

クスでユーザーが以前に操作した内容に基づいており、アプリが提供する候補よりも前に表示されます。検索履歴は無効にすることができます。

アプリの検索結果にフィルターまたはスコープを適用するために候補を使わないでください。フィルターとスコープのオプションは、検索結果を絞り込むことができるアプリの検索結果ページに結果と共に配置される必要があります。これに対して、候補は検索ボックスに配置される必要があります。これは、ユーザーが入力するクエリ テキストに候補が直接関係するためです。

クエリ候補

- 一般的に、ユーザーがすばやく検索できるようにクエリ候補を常に表示することを検討します。
- ユーザーがアプリで検索できるクエリ テキストをオート コンプリートする方法としてクエリ候補を使います。クエリ候補は、検索を実行するために必要な入力量を減らすことで、ユーザーがすばやく検索できるようにする効果的な方法です。ユーザーはクエリ全体を入力するのではなく、候補として表示されているクエリのいずれかを選んで検索をすぐに実行できます。
- ユーザーがクエリ候補を選んだら、選ばれたクエリの検索結果ページを即座に表示します。
- アプリのクエリ候補にユーザーの現在のクエリ テキストを含めます。クエリ候補は、現在のクエリ テキストに基づいてオート コンプリートされたテキストである必要があるため、現在のクエリ テキストが実際に含まれている必要があります。
- クエリ候補は、グローバル コンテキストの結果を提供するのではなく、アプリのコンテンツとアプリが提供できる結果を反映している必要があります。 アプリで提供できるコンテンツを検索することによって、ユーザーはクエリ候補からアプリで検索できる内容がわかります。たとえば、天気予報アプリで、ユーザーのクエリを自動的に完成させて、アプリが天気予報を提供できる都市を示します。

結果候補

- ユーザーのクエリに大部分が一致するか完全に一致する内容を提示する場合は結果候補を提供します。

- ユーザーが検索結果ページに移動しなくて済むように、結果候補を使って特定の結果の詳細を直接表示します。
- 結果候補は、適切な画像またはサムネイル、適切なタイトルまたはラベル、簡単な説明で構成する必要があります。画像サイズは 40x40 ピクセルに固定されています。
- 画像、タイトル、説明によって、ユーザーは結果候補が現在の検索対象かどうかをすばやく判断できるようになります。
- 結果セットが小さいため、アプリのキャンバスで結果をフィルター処理することを検討します。つまり、ユーザーが入力するときに、キーを押すたびに結果をすばやく更新する必要があります。これは、ユーザー入力時のフィルター処理の一種です。たとえば、リーディング リスト アプリでは、結果をこのようにフィルター処理します。検索履歴が無効になっている必要があります。そのアプリはキャンバスをできるだけ短時間で更新します。この機能の実装は、UI のパフォーマンスと、どれだけすばやく小さな結果セットを返すことができるかによって決まります。10 件の結果を取得するために、ユーザーが 3 文字を超える文字を入力する必要がある場合、既に説明したように候補を使うことを検討します。

区切り記号

- 類似する結果や概念的に近い結果のグループを定義するには、区切り記号を使います。
- 区切り記号はアプリによって検索ボックスに渡される候補の一覧に追加できますが、25 件の候補制限について各区切り記号がカウントされます。
- 複数の種類の結果候補を示す場合は、ユーザーが結果を区別できるように、ラベル付きの区切り記号を使います。たとえば、コンテンツの種類が異なる結果 (映画と TV 番組など) に対して複数の候補を示す場合は、ラベル付きの区切り記号を使って、"映画" と "TV 番組" のコンテンツの種類を区別できるようにします。

プレースホルダー テキスト

- ユーザーがアプリで検索できる内容を説明するために、検索ボックスでプレースホルダー テキストを使うことを検討します。たとえば、レシピ アプリでは、"レシピの検索" のように表示できます。
- 検索ボックスが空の場合のみプレースホルダー テキストを表示します。ユーザーが検索ボックスへの入力を開始したら、プレースホルダー テキストをクリアします。
- 検索ボックスのプレースホルダー テキストは、アプリの検索可能なコンテンツの簡単な説明に設定します。たとえば、アルバム名、曲名、アーティスト名による検索をサポートする音楽アプリの場合、プレースホルダー テキストを "アルバム名、アーティスト名、または曲名" に設定できます。
- プレースホルダー テキストを使って、検索対象のコンテキストをユーザーに伝えます。 これは、特定のコンテキストでアプリのサブセットが検索される場合に重要です。 たとえば、アーティストやアルバムのトピック専用のサブセクションでアーティストやアルバムを検索する音楽アプリでは、"アーティスト" や "アルバム" などのテキストを使い生ます。

検索結果ページのデザイン

ブランドを中心とした検索エクスペリエンスを作成します。検索結果のユーザー エクスペリエンス (UX) はコンテンツに合わせて調整する必要があります。これは、アプリでコンテンツを表示する方法と似ています。

ユーザーが検索クエリをアプリに送ると、クエリの検索結果を示すページが表示されます。アプリ用の検索結果ページをデザインすると、ユーザーにとって役立つ結果を適切なレイアウトで確実に表示できます。

検索結果ページにタイトルを付け、検索ボックスにはユーザーのクエリ テキストを表示してコンテキストを示します。

関連性の高い順から低い順に並べて配置して、検索結果のランクを示します。

検索結果がクエリに一致する理由を示す。

双方向の言語で検索アイコンをフリップしないでください。

候補のクエリに一致する部分を示すために一致する検索語句を強調表示します。一致するプロパティをユーザーに対して示している場合、あらゆる場面でこの機能が役立ちます。

結果のフィルター処理

- ユーザーが検索結果にフィルターまたはスコープを適用して検索結果を絞り込むことができるようにすると、アプリの検索結果ページの使い勝手を向上させることができます。
- クエリの結果の件数が多い場合は、結果をフィルター処理する方法をユーザーに提供することを検討します。
- ユーザーが検索結果ページから検索結果にフィルターまたはスコープを適用できるようにします。
- 各フィルターまたは各スコープで利用できる結果の数を示します。これにより、ユーザーは、検索が効果的に絞り込まれているかどうかを判断できます。
- フィルターを解除してすべての結果を表示する手段を用意します。

検索ボックスのサイズ変更

- アプリ内検索ボックス コントロールを使う場合、さまざまなウィンドウ サイズと、タイトルを基準にしたコントロールの位置を念頭に置きます。
- タイトルを切り詰めないようにしてください。
- ウィンドウ サイズが小さいときには、タイトル バーの内容に応じて、検索ボックスを折りたたんで検索アイコンを含むボタンにする場合があります。 ユーザーがボタンをクリックすると、検索ボックスが展開されて、タイトルなどの要素が隠されます。
- 戻るボタンが隠されないようにしてください。戻るボタンはナビゲーションの中心的な要素であり、ユーザーが常にアクセスできるようにすることが重要です。
- 検索候補のフライアウトの高さの最小値は 200 ピクセルです。
- 高さの既定の最大値は 300 ピクセルですが、この値は変更できます。

- 幅の最小値は 270 ピクセルです。この値は変更できますが、入力方式エディター (IME) ウィンドウの幅を 270 ピクセルより小さくすることはできません。

結果のページでは、常にクエリを事前設定しておきます。ユーザーが他のページに移動して戻ってきた場合、結果が保持されている必要があります。 結果ページからであっても、"進む" ナビゲーションではクエリ テキストを消去します。

プレースホルダーの最大文字数は 128 文字です。クエリ テキストの最大文字数は 2048 文字です。候補 (テキスト、詳細テキスト、画像の代替テキスト) のあらゆるテキスト フィールドの最大文字数は 512 文字です。

検索入力の有効化

検索ボックスを使っている場合は、検索ボックス コントロールをクリックまたはタップしなくても、単に文字を入力するだけで検索できるようにすることを検討します。これを行うには、**searchPane.showOnKeyboardInput** または **FocusOnKeyboardInput** を有効にします。これは、ユーザーが Windows 8 のスタート画面で使い慣れた種類の動作です。多くのユーザーは、キーボードを使って Windows 8 との対話操作を行います。ユーザーが入力による検索を行えるようにすると、キーボード操作の効率的な使用につながります。

アプリのすべてのページで検索入力を有効にすることは避けてください。ユーザーにとって検索入力が最も役立つ場所と、問題が発生する可能性のある場所を考慮します。たとえば、テキスト ボックスのように入力を受け取る他のコントロールにフォーカスが移動した場合は、検索入力を無効にします。無効にしないと、引き続き検索ボックスに文字が入力されます。

次のガイドラインを基にアプリに検索入力を追加する方法を把握し、使い勝手を高めてください。

- **アプリのメイン ページ (1 つまたは複数のページ) で検索入力を有効にする。** アプリのメイン ページ、およびアプリのすべての検索可能なコンテンツを直接的または間接的に表示する任意のページに対して、これを行います。メイン ページとは、ユーザーがアプリを開いたときに表示されるページです。 アプリには、同じコンテンツについてそれぞれ表示が異なる複数のメイン ページがある場合もあ

ります。その場合は、それらのページすべてで検索入力を有効にする必要があります。

- **アプリの検索結果ページで検索入力を有効にする。**
  検索結果を受け取った後に、もう一度検索したい場合もよくあります。検索結果のページから検索ボックスに直接入力できるようにし、キーボードのユーザーが新たな検索をすぐに実行できるようにしてください。
- **アプリのその他ほとんどのページで検索入力を無効にする。**
  ほとんどの場合、アプリのメイン ページ以外のページまたは検索結果のページ以外のページでは検索ボックスに直接入力できないようにする必要があります。1 つ以上の入力ボックスを持つページでは、テキスト ボックスにフォーカスが移動した場合にアプリで検索入力を無効にできます。

検索コントラクトの使用

アプリが検索コントラクトを使う場合は、次のガイダンスに従ってください。

- 検索アイコン付きのボタンをアプリに追加し、このボタンによって検索ウィンドウをプログラムから呼び出して、スコープを適切に設定します。検索グリフを使ってウィンドウを呼び出します (15pt の Segoe UI Symbol 0xE0094)。
- 目立つ検索アイコンを使うことで、どこから開始すればよいかをユーザーに対してビジュアルを明確に示すことができます。ユーザーがこのアイコンを選ぶと、アプリのプログラムによって検索チャームが開き、ユーザーは検索ウィンドウを使ってクエリを入力できるようになります。検索チャームをこのように使うことによって、アプリがさらに直感的になります。
- アプリで検索ボックスを使う場合は、**SearchPane** API を呼び出せなくなります。この API を呼び出すと、"**Windows.UI.Xaml.Controls.SearchBox**" 型のインスタンスを作成できないことを示すメッセージと共に、例外がスローされます。アプリでは引き続き検索コントラクトを使えます。実際、この検索コントラクトを使って検索のアクティブ化コードをサポートします。

スクロール バーのガイドライン

説明

パンとスクロールを行うと、画面の境界外のコンテンツを拡張表示することができます。

スクロール ビューアー コントロールは、ビューポート内に適合する量のコンテンツと、一方または両方のスクロール バーからなります。パンとズームのためにタッチ ジェスチャを使うことができ (操作中にのみスクロール バーはフェードインします)、またスクロールのためにポインターを使うことができます。フリック ジェスチャでは、慣性を伴ってパンします。

Windows: 検出された入力デバイスに基づいて、次の 2 種類のパン表示モードが使われます。パン インジケーター (タッチを使う場合) とスクロール バー (マウス、タッチパッド、キーボード、スタイラスなど、その他の入力デバイスを使う場合) です。

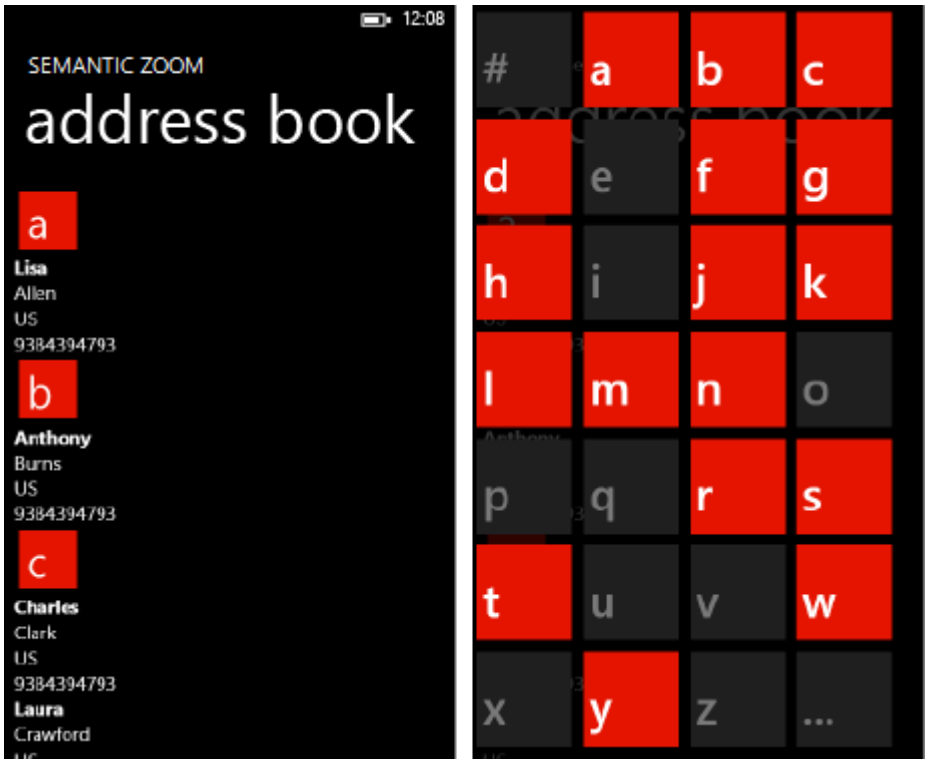
例

推奨と非推奨

- パン インジケーターとスクロール バーを表示して、位置とサイズがわかるようにします。これらのコントロールは、カスタム ナビゲーション機能がある場合には非表示にします。
- コンテンツ領域が 1 つのビューポート境界 (垂直方向または水平方向) を超えている場合は、単一軸のパンを使います。コンテンツ領域が両方のビューポート境界 (垂直方向と水平方向) を超えている場合は、2 軸のパンを使います。
- リスト ボックス、ドロップダウン リスト、テキスト入力ボックス、グリッド ビュー、リスト ビュー、ハブ コントロールの組み込みのスクロール機能を使います。こうしたコントロールでは、同時に表示する項目が多すぎる場合に、ユーザーが項目のリストを水平方向、垂直方向のいずれかにスクロールできます。
- ユーザーがより大きな領域の周囲で両方向にパンすること、そしておそらくズームできるようにする場合、たとえばユーザーが (画面に適合するサイズに設定されたイメージではなく) フル サイズのイメージをパンおよびズームできるようにする場合には、スクロール ビューアー内にイメージを配置します。
- ユーザーが長いテキスト パスをスクロールする場合、垂直方向にのみスクロールするようにスクロール ビューアーを構成します。
- 1 つのオブジェクトのみを含める場合にスクロール ビューアーを使います。1 つのオブジェクトをレイアウト パネルとし、その任意の数のオブジェクトを含めることができる点に注意してください。

セマンティック ズームのガイドライン

Windows アプリ：セマンティック ズーム コントロールの拡大表示と縮小表示



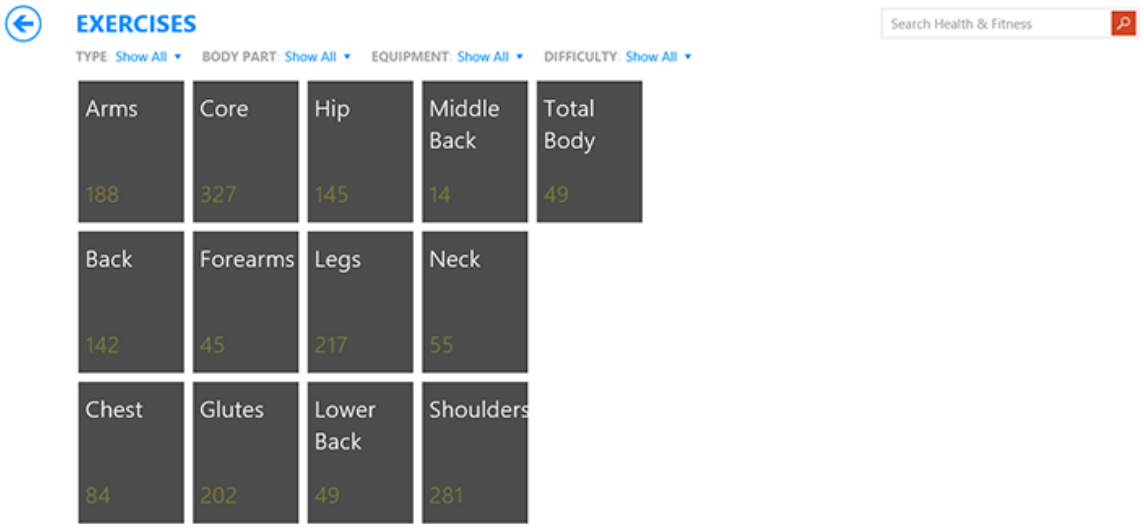
Windows Phone アプリ：セマンティック ズーム コントロールの拡大表示と縮小表示

説明

セマンティック ズーム コントロールを使うと、ユーザーは同じデータ セットの 2 つの異なるセマンティック ビューの間でズームを実行できるようになります。一方の表示にはキーでグループ化された項目の一覧が含まれ、もう一方の表示にはグループ キーの一覧が含まれます。グループ キーをタップすると、そのグループの項目がもう一度拡大表示されます。

データ セットが、それぞれ効果が得られる体の部位でグループ化された運動の一覧の場合、拡大表示では体の部位 (項目) でグループ化された運動の一覧が表示され、縮小表示では体の部位 (グループ キー) の一覧が表示されます。大きなデータ セットを使う場合、縮小表示してすべてのグループ キーを 1 ページで確認できるようにすると、ユーザーは、そうしなかった場合に確認のために何回もスクロールする離れた項目にもすばやく移動できます。

例



推奨と非推奨

- 使用可能または操作可能な要素のタッチ ターゲットを適正なサイズにする。詳しくは、「ターゲットの設定のガイドライン」をご覧ください。
- セマンティック ズームの範囲を適切で直観的なものにする。ユーザーは、多くの場合、表示された項目を囲む領域内からセマンティック ズームを開始します。 セマンティック ズームの範囲は、その領域を余裕をもって囲める大きさにしてください。たとえば、Windows ストアには、アプリ リストの周りに十分な余白があり、ユーザーはその場所に指を置いて拡大や縮小を行うことができます。
- 各ビューに固有の構造とセマンティックを使う。

    - グループ化したコレクションの項目にはグループ名を使います。
    - グループ化せずに並べ替えたコレクションには並べ替え順序 (日付の場合は時系列順、名前の場合はアルファベット順など) を使います。
    - ドキュメントのコレクションを表すにはページを使います。
- ズーム レベルごとに項目のレイアウトとパン方向が変わらないようにする。レイアウトとパン操作は、ズーム レベルに関係なく一貫して予測できるものにしてください。
- 縮小モードのページ (画面) の数は 3 つまでにする。セマンティック ズームを使うと、ユーザーはすばやくコンテンツに移動できます。パンが多すぎると、この利点が失われます。
- セマンティック ズームを使ってコンテンツの範囲を変更しない。たとえば、フォト アルバムをファイル エクスプローラーのフォルダー表示に切り替えないでください。
- セマンティック ズーム コントロールの子コントロールには境界線を設定しない。セマンティック ズーム コントロールとその子コントロールの両方に対して境界線を設定すると、セマンティック ズーム の境界線と、その中に表示される子コントロールの境界線の両方が表示されます。拡大表示または縮小表示を実行すると、子コントロールの境界線がコンテンツに合わせて拡大または縮小されるため、見た目が良くありません。境界線は セマンティック ズーム コントロールだけに設定してください。

その他の使い方のガイダンス

光学式ズームでは、コンテンツの拡大縮小と倍率が変更されます。セマンティック ズームでは、キーでグループ化された項目とこのようなキーの一覧との間で、表示されるコンテンツのセマンティクス (意味) が切り替えられます。表示するグループ化されたデータの一覧が長い場合 (たとえば、体の部位でグループ化された運動やイニシャルでグループ化された名前) は、セマンティクス ズーム コントロールを使うことを検討してください。

'A' で始まる名前から 'Z' で始まる名前までスクロールするには、何度もスワイプ ジェスチャを行う場合があります。セマンティック ズーム コントロールを使うと、わずか 2 回のタ

ップを行うだけです。1 回目のタップで縮小表示してイニシャルの一覧が表示され、2 回目のタップでもう一度拡大表示して 'Z' が表示されます。

セマンティック ズームでは、その 2 つのビュー間でズームを管理します。ビューはリスト ビューまたはグリッド ビューで、状況に応じて項目またはグループ キーが表示されます。

セマンティック ズームでは、コンテンツを整理して表示するために 2 つの分類モード (ズーム レベル) を使います。低レベル (拡大) モードは、通常、すべての項目をフラットな構造で表示する場合に使います。一方、高レベル (縮小) モードでは、項目がグループ別に表示され、ユーザーはすばやくコンテンツをナビゲートして閲覧できます。

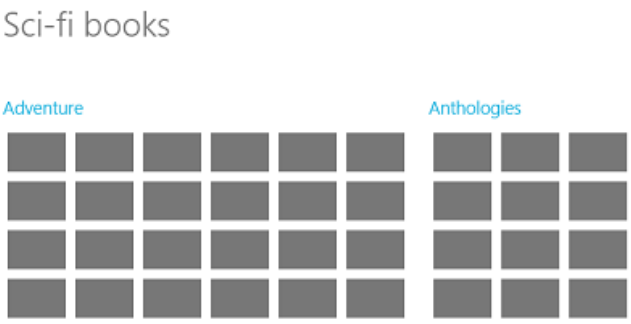
セマンティック ズーム操作は、ピンチ ジェスチャとストレッチ ジェスチャ (指を広げて拡大、互いに近づけて縮小)、Ctrl キーを押しながらマウスのスクロール ホイールをスクロール、あるいは Ctrl キーを (テンキーがない場合は Shift キーも同時に) 押しながらプラス (+) キーまたはマイナス (-) キーを押すことで実行されます。

セマンティック ズームの概要については、このビデオをご覧ください。

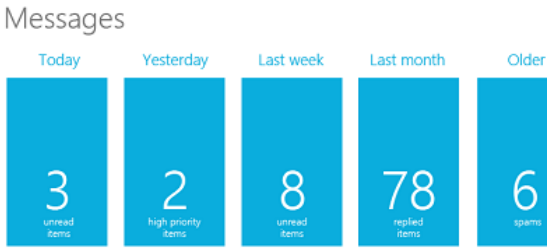
セマンティック ズームは、たとえば次のようなアプリで使うことができます。

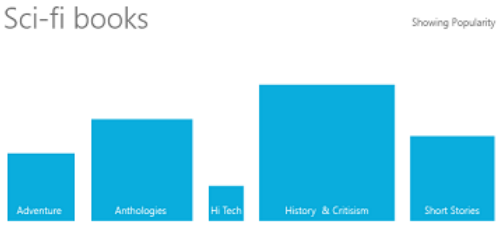
- 連絡先をアルファベット順に (またはその他の何らかの方法で) まとめた、アルファベットの文字を使ってデータを表示するアドレス帳。ユーザーが文字を選んで拡大すると、その文字に関連付けられた連絡先が表示されます。
- 画像をメタデータ (撮影日など) 別にまとめたフォト アルバム。ユーザーが特定の日付を選んで拡大すると、その日付に関連付けられた一連の画像が表示されます。
- 項目をカテゴリ別にまとめた製品カタログ。
- セマンティック ズームを使って、他にも次のようなレイアウトが可能です。

拡大表示



縮小表示





セマンティック ズームを使ったナビゲーション

パンとスクロールだけでもコンテンツのナビゲーションはできますが (「パンのガイドライン」を参照)、それらをセマンティック ズームと組み合わせると、高度なナビゲーションや整理が可能になります。

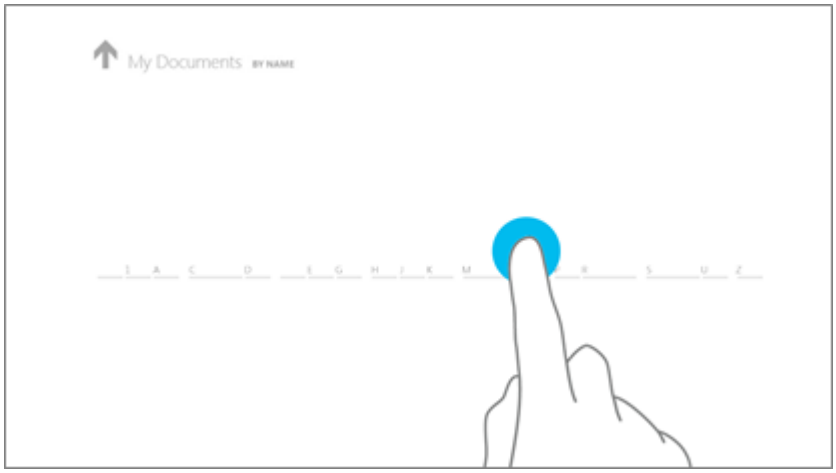
パンとスクロールは、コンテンツが少なく、移動距離が短い場合に便利です。ただし、コンテンツが多いと、すばやいナビゲーションが難しくなります。セマンティック ズームを使うと、大量のコンテンツをナビゲートするときに距離が長いと感じることがはるかに少なくなり、コンテンツ内の位置にすばやく簡単にアクセスできます。

注 セマンティック ズームと光学式ズーム (「光学式ズームとサイズ変更のガイドライン」を参照) を混同しないように気を付けてください。操作方法と基本的な動作 (ズーム係数に基づいて詳しく表示したり簡単に表示したりする動作) は同じですが、光学式ズームでは、コンテンツ領域またはオブジェクトの倍率調整を写真のように行います。

スクロール移動

次の 3 つの図で示すように、縮小モードでコンテンツをタップすると、拡大表示に切り替わり、タップしたポイントにパンされます。

縮小され、コンテンツ全体がタッチ ターゲットになっています。



コンテンツのセクションをタップします。

拡大表示に切り替わり、タップした領域にパンされます。

切り替え

セマンティック ズームのレベルの切り替えには、スムーズなクロスフェードと拡大縮小のアニメーションが使われます。これは Windows タッチの既定の動作であり、カスタマイズすることはできません。

考慮事項と推奨事項

アプリの 2 つのセマンティック レベルは開発者が定義します。

縮小モードを設計する際には、次の点を考慮してください。

- ズーム レベルごとに情報の構造や表示をどのように変えるか。

- データをナビゲートするときにヒント ("道標") があったら役立つかどうか。
- どれくらいの量のコンテンツであれば、セマンティック ビューを活用してパンやスクロールを最小限に抑えられるか。

これらについて考慮するうちに、互いに矛盾することがよくあります。移動先の情報を多くしてコンテンツを充実させたいと思っても、この情報の量はセマンティック レベルの全体の長さとのバランスを考えて決めなければなりません。縮小モードでユーザーが何度もパンしていては、セマンティック ズームの最大の利点が失われてしまいますーつまり、すばやいナビゲーションを実現できません。

Windows 8 のハンズオン ラボ

セマンティック ズームなど Windows 8 の主要機能を試してみたい場合は、Windows 8 のハンズオン ラボをダウンロードしてください。このラボでは、任意のプログラミング言語 (JavaScript と HTML または C# と XAML) で Windows ストア アプリのサンプルを作成する方法をモジュール形式の手順で紹介します。

外観と操作

セマンティック ズーム コントロールの外観は、どちらのビューが表示されているかによって異なります (それぞれのビューはリスト ビュー コントロールまたはグリッド ビュー コントロールになります)。セマンティック ズームは、拡大表示した場合にキー ヘッダーでグループ化される項目の一覧として表示され、縮小表示した場合にキーの一覧として表示されます。

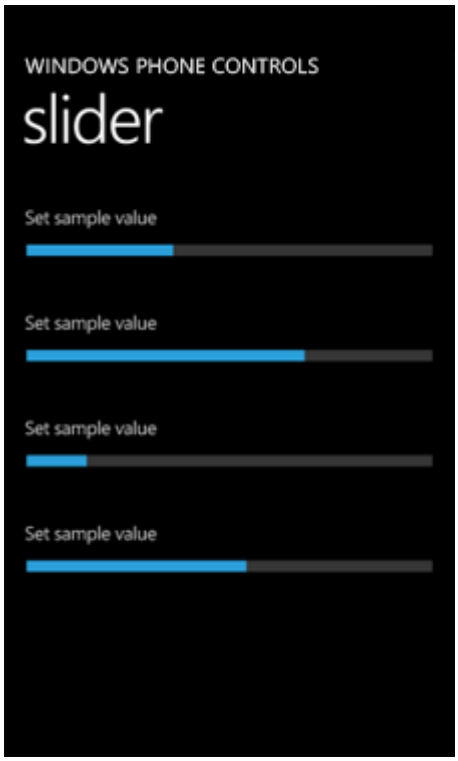
拡大表示中に項目をタップすると、項目が選択されるか、項目の詳細ページに移動します。縮小表示中にグループ キーをタップすると、スクロールによって表示されたそのグループで拡大表示されます。

Windows: ズーム操作は、ピンチ ジェスチャとストレッチ ジェスチャ (指を互いに近づけて縮小、離して拡大)、Ctrl キーを押しながらマウスのスクロール ホイールをスクロール、あるいは Ctrl キーを (テンキーがない場合は Shift キーも同時に) 押しながらプラス (+) キーまたはマイナス (-) キーを押すことで実行されます。

Windows Phone: 拡大表示中にグループ キー ヘッダーをタップすると縮小表示されます。

スライダーのガイドライン

Windows アプリ：スライダー コントロール

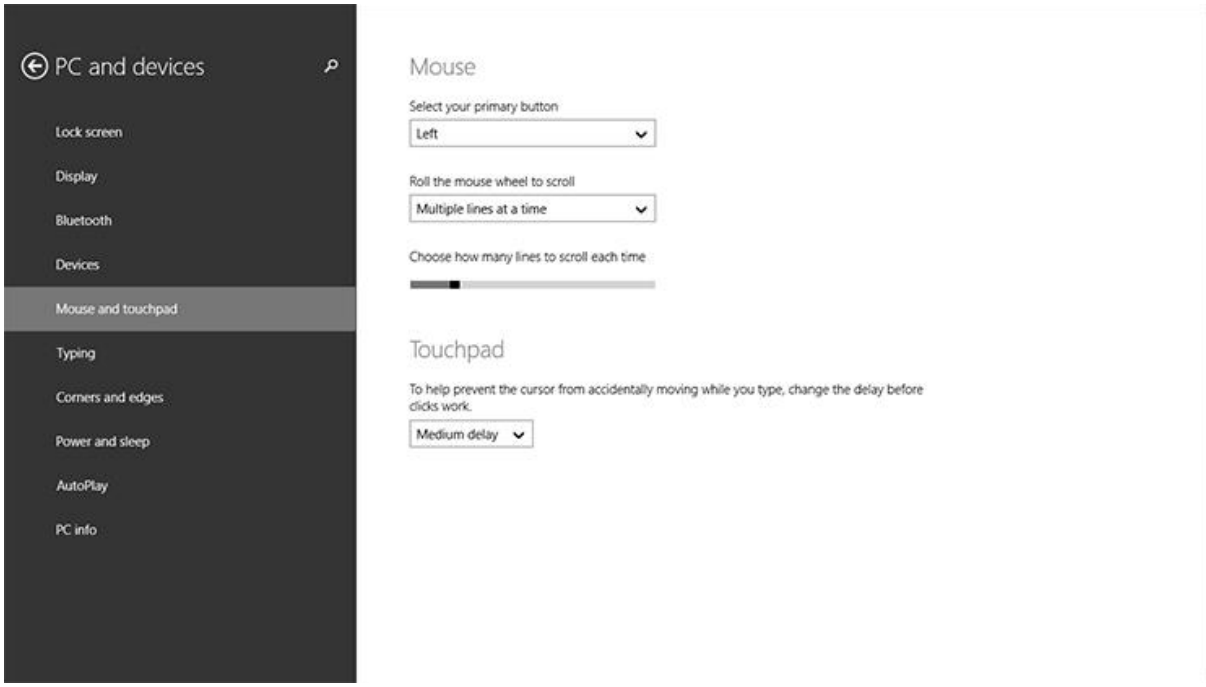


Windows Phone アプリ：スライダー コントロール

説明

スライダー コントロール (範囲コントロールとも呼ばれます) では、ユーザーがトラック上をタップするか前後にスクラブして、有効な範囲から値を設定できます。

例



適切なコントロールの選択

定義された連続的な値 (音量や明るさなど) または個別の値の範囲 (画面解像度の設定など) をユーザーが設定できるようにする場合に、スライダーを使います。

スライダーは、ユーザーが値を数値でなく相対的な量であると考えている場合に適しています。たとえば、ユーザーはオーディオの音量を数値の 2 や 5 ではなく、低や中に設定しようと考えます。

二者択一の設定には、スライダーは使いません。代わりに、トグル スイッチ を使います。

スライダーを使うかどうかを決める際には、他にも次のような点を考慮します。

- **設定が相対的な量のように見えるか?** 見えない場合は、ラジオ ボタンまたは 選択コントロールを使います。
- **設定は正確な既知の数値か?** そのような数値である場合は、数値テキスト ボックスを使います。
- **設定の変更による効果をすぐに確認できると、ユーザーにとって便利か?** 便利である場合は、スライダーを使います。たとえば、色合い、鮮やかさ、明度の値を変更した場合の効果をすぐに確認できると、ユーザーは色をより簡単に選べるようになります。

- **設定に 4 つ以上の値の範囲があるか?**　ない場合は、ラジオ ボタンを使います。
- **ユーザーが値を変えられるか?**　スライダーは、ユーザーが操作するためのものです。ユーザーが値を変えられない場合は、代わりに読み取り専用のテキストを使います。

スライダーと数値テキスト ボックスのどちらを使うかを決める際に、次の場合には数値テキスト ボックスを使います。

- 画面領域が狭い。
- ユーザーがキーボードを使おうとする可能性が高い。

次の場合にはスライダーを使います。

- ユーザーにとって、すぐに結果がわかると便利。

推奨と非推奨

- コントロールのサイズは、ユーザーが値を簡単に設定できる大きさにします。 個別の値を設定する場合は、ユーザーがマウスを使って値を簡単に選べるようにします。スライダーのエンドポイントが、常にビューの境界内にあることを確認します。
- ユーザーが選んでいるときまたは選んだ後で、すぐに結果が確認できるようにします (それが実際的な場合)。たとえば、Windows のボリューム コントロールは、選ばれたオーディオ音量を示すためにビープ音を鳴らします。
- 値の範囲を示すためにラベルを使います。 例外: スライダーが垂直方向で、上部のラベルが最大、高、多などの場合、下部の意味は明らかであるため、ラベルを省略できます。
- スライダーを無効にする場合は、関連するすべてのラベルまたはフィードバックのビジュアル効果も無効にします。
- スライダーのフロー方向や向きを設定するときには、テキストの方向を考慮してください。言語によって、左から右に書く場合と、右から左に書く場合があります。
- スライダーは、進行状況インジケーターとしては使いません。
- スライダーのつまみは、既定のサイズのままにします。

- 値の範囲が広く、ユーザーが選ぶのは範囲内のいくつかの代表的な値であることがほとんどの場合は、連続的なスライダーを作成しません。 代わりに、それらの値だけを、許可される間隔として使います。たとえば、時間の最大値は 1 か月であっても、ユーザーが 1 分、1 時間、1 日、1 か月のいずれかを選ぶ場合は、4 つの間隔位置だけのスライダーを作成します。

その他の使い方のガイダンス

適切なレイアウトの選択: 水平または垂直

スライダーは、水平方向または垂直方向にレイアウトできます。次のガイドラインを使って、使用するレイアウトを決めます。

- 自然な方向を使います。たとえば、スライダーが現実世界の値を表していて、通常は垂直方向に表示される場合 (気温など) は、垂直方向にします。
- 動画アプリのように、メディア内をシークするためにコントロールが使われる場合は、水平方向にします。
- 1 つの方向 (水平または垂直) にパンするページでスライダーを使う場合は、パンの方向とは異なる方向をスライダーに使います。 そうしないと、ユーザーはページをパンしようとしてスライダーをスワイプし、誤って値を変えてしまう場合があります。
- 使用する方向がまだ決まらない場合は、ページ レイアウトに適した方を使います。

範囲方向に関するガイドライン

範囲方向とは、現在の値から最大値へスライダーを動かす方向のことです。

- 垂直方向スライダーでは、読みの方向に関係なく、最大値をスライダーの上部に配置します。たとえば、音量スライダーでは、最大の音量設定を常にスライダーの上部に配置します。他の種類の値 (曜日など) では、ページの読みの方向に従います。
- 水平方向のスタイルでは、ページ レイアウトが左から右への場合は、低い方の値をスライダーの左側に配置します。ページ レイアウトが右から左への場合は、右側に配置します。

- 前のガイドラインの 1 つの例外は、メディア シーク バーです。このバーでは、低い方の値を常にスライダーの左側に配置します。

間隔と目盛りのガイドライン

- スライダーで最小値から最大値までの任意の値を許可するのではない場合は、間隔位置を使います。たとえば、購入する映画のチケットの数を指定するためにスライダーを使う場合、浮動小数点値は許可しません。間隔の値を 1 にします。
- 間隔(スナップ位置とも言います) を指定する場合、最後のステップがスライダーの最大値に揃うようにします。
- 主な、または重要な値の位置をユーザーに示す場合は、目盛りを使います。たとえば、ズームを制御するスライダーでは、50%、100%、200% の目盛りを設定します。
- 設定のおおよその値をユーザーが知る必要がある場合に、目盛りを表示します。
- 選択した設定の正確な値を、コントロールを操作しなくてもユーザーが確認できるようにするには、目盛りと値ラベルを表示します。または、値のツールチップを使って、正確な値が見られるようにします。
- 間隔位置がわかりにくい場合は、常に目盛りを表示します。たとえば、スライダーの幅が 200 ピクセルで、200 のスナップ位置がある場合は、ユーザーはスナップの動作に気付かないので、目盛りを非表示にできます。しかし、スナップ位置が 10 個しかない場合は、目盛りを表示します。

ラベルのガイドライン

- **スライダー ラベル**

  スライダー ラベルは、スライダーの使用目的を示します。
  - ラベルの末尾には句点を付けません。
  - スライダーのあるフォームで、ほとんどのラベルがコントロールの上にある場合は、ラベルをスライダーの上に配置します。
  - スライダーのあるフォームで、ほとんどのラベルがコントロールの横にある場合は、ラベルをスライダーの横に配置します。
  - ユーザーがスライダーにタッチするときに、指でラベルが見えなくなる場合があるので、ラベルをスライダーの下には配置しません。

- **範囲ラベル**

  範囲 (容量) ラベルは、スライダーの最小値と最大値を示します。

  - 垂直方向であることによって明白である場合以外は、スライダーの範囲の両端をラベルに表示します。
  - 各ラベルは、できれば 1 ワードだけにします。
  - 末尾に句点を付けません。
  - これらのラベルは、説明的で対比的なものにします。例: 最大/最小、多/少、低/高、小/大

- **値ラベル**

  値ラベルは、スライダーの現在の値を表示します。

  - 値ラベルが必要な場合は、スライダーの下に表示します。
  - テキストをコントロールに対して中央に配置し、単位 (ピクセルなど) を付記します。
  - スライダーのつまみはスクラブ中に隠れるため、ラベルや他のビジュアル効果で現在の値を表示することをお勧めします。スライダー設定のテキスト サイズは、スライダー以外の適切なサイズのサンプル テキストに連動させることができます。

外観と操作

スライダーはトラックとつまみで構成されます。トラックは入力できる値の範囲を表すバーです (オプションでさまざまなスタイルの目盛りを表示できます)。つまみは、ユーザーがトラックをタップするか、トラックを前後にスクラブして位置を調整できるセレクターです。

スライダーには大きなタッチ ターゲットが設定されています。タッチのアクセシビリティを維持するには、スライダーを表示の端から十分に離して配置する必要があります。

カスタム スライダーを設計する際は、余分な要素をできるだけなくし、ユーザーに必要なすべての情報を示す方法を検討してください。ユーザーが設定を理解できるように単位を表示する必要がある場合は、値ラベルを使います。これらの値を視覚的に示す方法を工夫してください。たとえば、音量を調整するスライダーでは、スライダーの最小の端に音波のないスピーカーのグラフィック、最大の端に音波のあるスピーカーのグラフィックを表示できます。

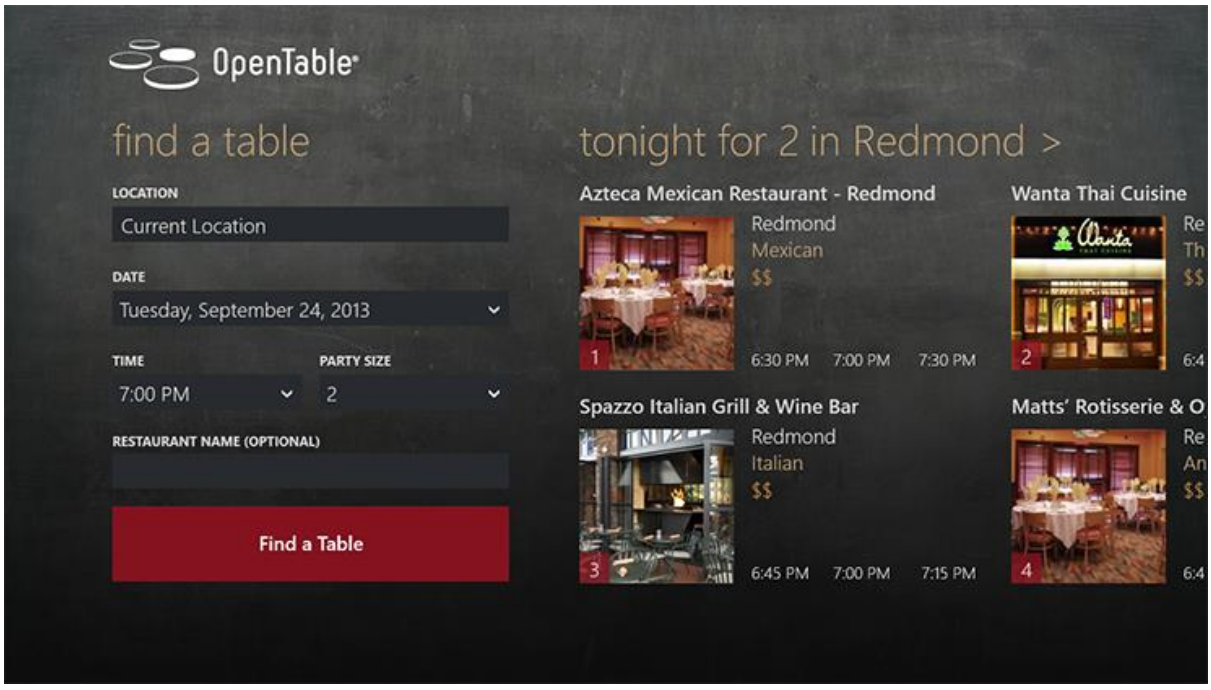
TimePicker のガイドライン

説明

TimePicker は、ユーザーがタッチ、マウス、またはキーボード入力を使ってローカライズされた時刻を選択できる標準化された方法です。

ユーザーが単一の時刻を選ぶ必要がある場合は、TimePicker を使います。TimePicker は、使われる画面領域が固定されていて選択の数とは関係がないため、領域を節約するうえで優れたオプションです。

例



推奨と非推奨

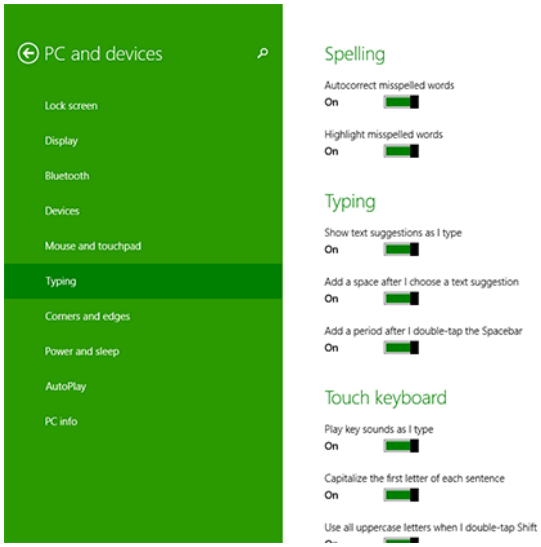
- ユーザーが単一の時刻を選ぶ必要がある場合は、TimePicker を使います。
- 領域を効率的に使う必要があるときには、TimePicker を使って時刻を選択できるようにします。
- 現在の時刻の表示には TimePicker を使わないでください。 これはユーザーによって設定されることを意図した静的な表示です。
- コマンドを実行する場合、ダイアログ ボックスなどの別のウィンドウを表示する場合、または別のコントロールを動的に表示する場合には、TimePicker を使わないでください。

トグル スイッチのガイドライン

説明

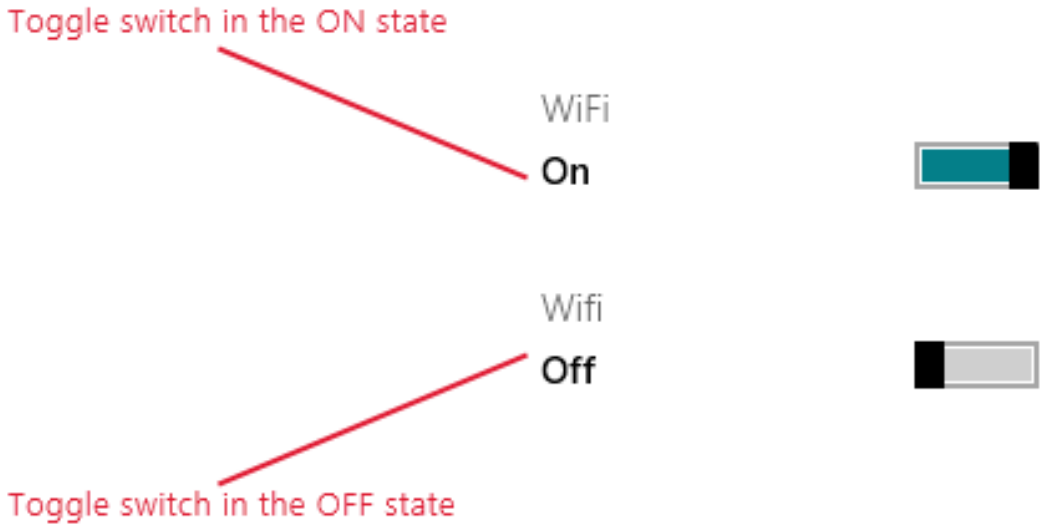
トグル スイッチは、ユーザーが項目をオンまたはオフに切り替えることができる物理的なスイッチを模したものです。 このコントロールには、オン (checked が **true**) とオフ (checked が **false**) の 2 つの状態があります。Windows ストア アプリにトグル スイッチ コントロールを追加するには、次のガイドラインに従ってください。

例



適切なコントロールの選択

トグル スイッチは、ユーザーが変更した後すぐに有効になるバイナリ操作に対して使います。たとえば、トグル スイッチを使って、サービスまたはハードウェア コンポーネントをオンまたはオフにできます。



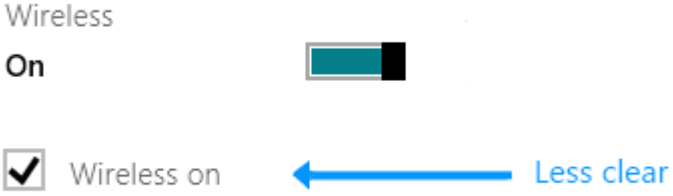
トグル スイッチを使うかどうかを判断するための良い方法は、そのような操作を行うために物理的なスイッチを使うかどうかを考えてみることです。

ユーザーがスイッチをオンまたはオフに切り替えた後は、対応する操作をすぐに実行します。

トグル スイッチとチェック ボックスの選択

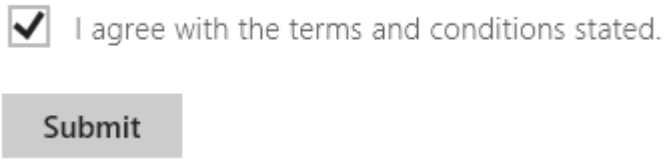
状況によっては、トグル スイッチとチェック ボックスのどちらでも使える場合があります。次のガイドラインに従って、どちらかを選択します。

- ユーザーが変更した後すぐに変更が有効になるようなバイナリ設定に対しては、トグル スイッチを使います。

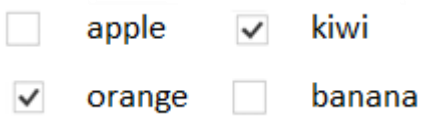


トグル スイッチの場合は、ワイヤレスがオンになっていることが明らかです。一方、チェック ボックスの場合は、ワイヤレスが現在オンになっているのか、またはオンにするためにチェック ボックスをオンにする必要があるのか、ユーザーが考える必要があります。

- 変更を有効にするためにユーザーが追加の手順を実行する必要があるときは、チェック ボックスを使います。たとえば、ユーザーが [送信] や [次へ] などのボタンをクリックして変更を適用する必要がある場合は、チェック ボックスを使います。



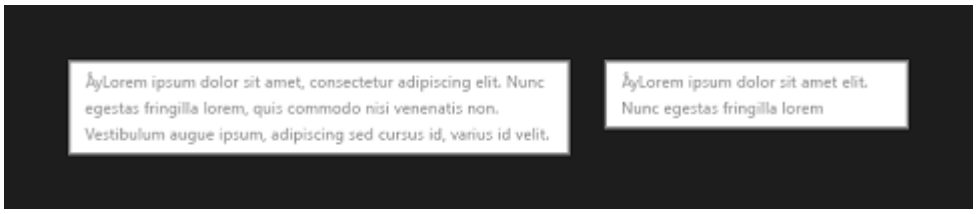
- ユーザーが複数の項目を選択できるときは、チェック ボックスまたはリスト ビューを使います。



推奨と非推奨

- 設定に対して具体的なラベルがある場合は、それらを "オン" や "オフ" の代わりに使います。特定の設定に対してより適切な、対になる項目を表す短い (3 ～ 4 文字の) ラベルがある場合は、それらを使います。たとえば、設定が "画像の表示" である場合は、"表示/非表示" などを使います。より具体的なラベルを使うと、UI のローカライズ時に役立ちます。
- 必要な場合以外は、"オン" または "オフ" のラベルを変更しない。設定に対してより具体的なラベルがない場合は、既定のラベルを使う必要があります。
- 2 ～ 3 文字より長いラベルを使わない。

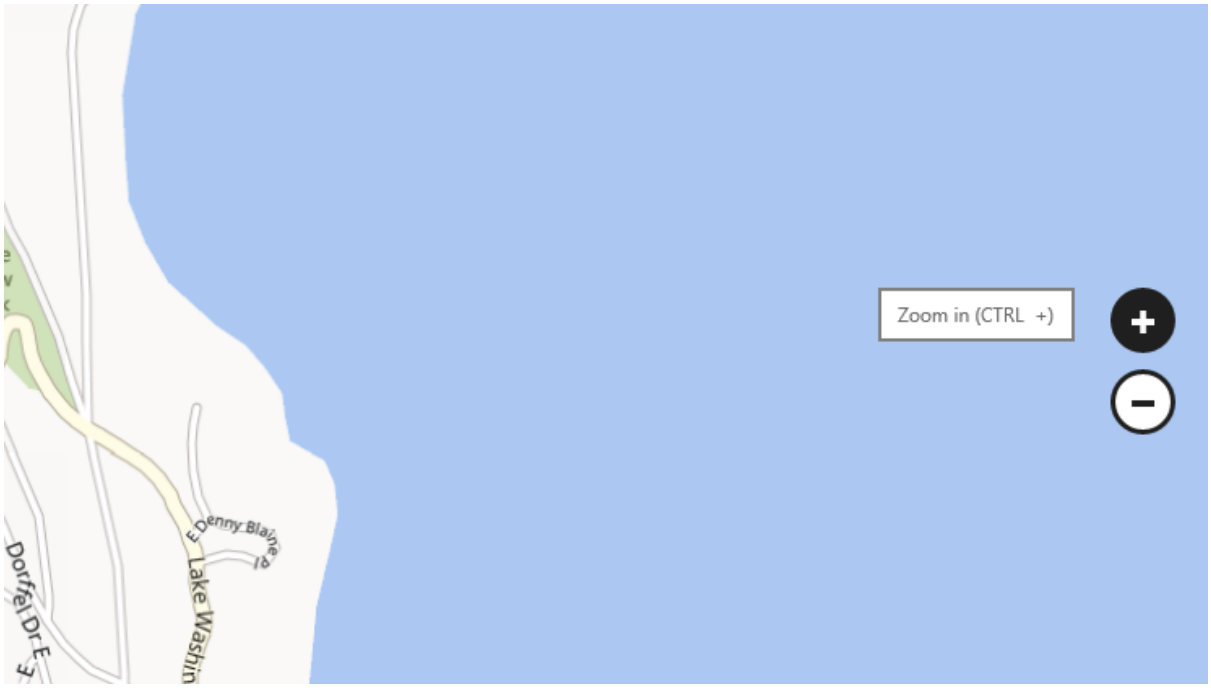
ツールチップのガイドライン



説明

ツールチップは、他のコントロールまたはオブジェクトにリンクされた短い説明です。ツールチップを使うと、UI では直接説明されていない、なじみのないオブジェクトをユーザーが理解しやすくなります。ツールチップは、ユーザーがコントロール上で長押しするか、マウス ポインターをコントロール上にホバーすると、自動的に表示されます。ツールチップは、ユーザーが指、マウス ポインター、またはペン ポインターを移動したときに消えます。

例



適切なコントロールの選択

ユーザーに操作の実行を指示する前に、ツールチップを使ってコントロールに関する詳しい情報を表示します。また、ツールチップを使うと、タッチ ダウン中に指の下に項目を表示できるため、ユーザーは自分がどこをタッチしているのかを知ることができます。ただし、最初は他の方法で場所を明確にするよう試みてください。たとえば、より大きなコントロールやより大きな領域を使ったり、コントロールのアクティブ状態やホバー状態にスタイルを適用したりします。

ツールチップはどのような場合に使えばよいでしょうか。それを判断するには、以下の質問を考えます。

- **情報はポインターをホバーすることで表示されますか?**
  そうでない場合は、別のコントロールを使います。ツールチップを、ユーザーの操作の結果としてのみ表示します。自動的には表示しません。
- **コントロールにはテキスト ラベルがありますか?**
  ない場合は、ツールチップを使ってラベルを表示します。ほとんどのコントロールにはラベルを付けることをお勧めします。それらのコントロールには、ツールチップは必要ありません。グラフィックのラベルを持つツール バー コントロールやコマンド ボタンには、ツールチップが必要です。
- **より詳しい説明や追加情報がオブジェクトに対して役立ちますか?**
  そうであれば、ツールチップを使います。ただし、このテキストは、主要なタスクに必須なものではなく、補助的なものである必要があります。必須なものであれば、直接 UI に配置して、ユーザーが探さなくても済むようにします。
- **表示する補助的な情報は、エラー、警告、または状態ですか?**
  その場合は、フライアウトなど、他の UI 要素を使います。
- **ユーザーがツールチップを操作する必要がありますか?**
  その場合は、別のコントロールを使います。ツールチップはマウスを動かすと消えるため、ユーザーはツールチップを操作できません。
- **ユーザーが補助的な情報を印刷する必要がありますか?**
  その場合は、別のコントロールを使います。
- **ユーザーがツールチップを煩わしいと感じますか?**

その場合は、別の手段を使うことを検討します。何もしない、という選択肢もあります。煩わしいと感じる可能性があってもツールチップを使う場合は、ユーザーがツールチップをオフにできるようにします。

ツールチップの適切な使い方の例を次に示します。

- ユーザーがカレンダーの日付をタッチしたときに曜日を表示する。
- ユーザーがハイパーリンクをタッチしたときにリンク先の Web サイトのプレビューを表示する。

推奨と非推奨

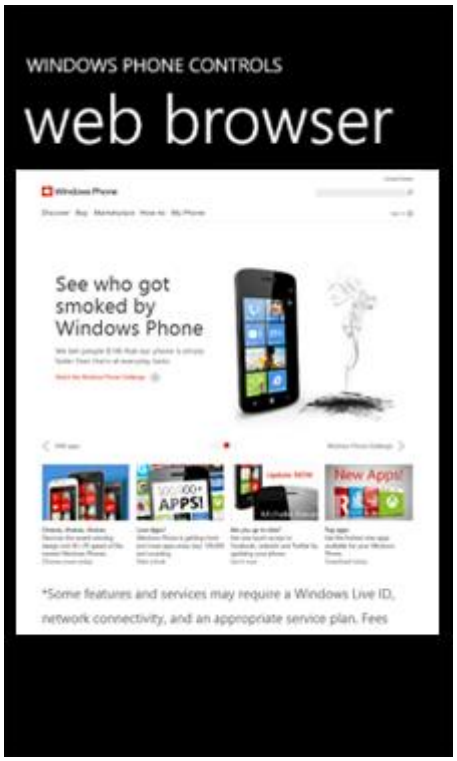
- ツールチップは慎重に使う (または使わない)。ツールチップは作業の中断になります。ツールチップはフライアウトと同じように煩わしい場合があるため、大きな付加価値がない限り使わないでください。
- ツールチップのテキストは簡潔なものにする。ツールチップは短い文やフレーズに適しています。大きなテキストのまとまりは読みにくく、圧迫感を与えます。
- 役に立つ補足的なツールチップ テキストを作成する。ツールチップのテキストは、情報として役に立つ必要があります。表示しなくても明らかな情報や、既に画面上に表示されている内容の繰り返しなどは避けます。ツールチップのテキストは常に表示されているわけではないため、ユーザーが必ずしも読まなくても問題がないような、補足的な情報である必要があります。重要な情報は、名前から判別できるコントロール ラベルを使うか、補足的なテキストを適切な場所に配置することで伝えるようにします。
- 状況に応じて画像を使う。ツールチップ内に画像を使うとよい場合もあります。たとえば、ユーザーがハイパーリンクをタッチしたときに、ツールチップを使ってリンク先ページのプレビューを表示できます。
- 既に UI に表示されているテキストは、ツールチップとして表示しない。たとえば、ボタンをタッチするとテキストが隠れる場合を除き、同じテキストが表示されているボタンにはツールチップを表示しません。
- ツールチップ内に対話的なコントロールを配置しない。
- 対話的に見えるような画像をツールチップ内に配置しない。

その他の使い方のガイダンス

ツールチップは慎重に使い、タスクを完了しようとしているユーザーにとって明らかに重要である場合にのみ追加します。1 つの目安は、情報が同じエクスペリエンスのどこかで入手できる場合、ツールチップは必要ありません。価値あるツールチップによって、不明瞭な操作を明確にします。

ユーザーに操作の実行を指示する前に、ツールチップを使ってコントロールに関する詳しい情報を表示します。また、ツールチップを使うと、タッチ ダウン中に指の下に項目を表示できるため、ユーザーは自分がどこをタッチしているのかを知ることができます。

Web ビューのガイドライン



Windows Phone アプリ：Web ビュー コントロール

説明

Web ビュー コントロールは、Internet Explorer のように動作するビューをアプリに組み込みます。また Web ビュー コントロールでは、ハイパーリンクの表示と動作が可能です。

推奨と非推奨

- フォーム ファクターに対して適切なテキスト サイズにします。たとえば、Windows Phone の最小値は 15 ポイントです。
- 読み込まれた Web サイトがデバイスに対して正しく書式設定されており、アプリの他の部分に対して一貫性のある色、タイポグラフィ、ナビゲーションが使用されていることを確認します。詳しくは、「Windows Phone 用の Web 開発」をご覧ください。
- ユーザーはテキスト入力時に拡大できることに気が付かない場合があるために、入力フィールドは適切なサイズに設定する必要があります。
- Web ビューがアプリの他の部分とは異なって見える場合は、関連タスクを実行するための代替のコントロールまたは手段を検討します。Web ビューがアプリの他の部分に一致すれば、ユーザーにはすべてが 1 つのシームレスなエクスペリエンスとして認識されます。

その他の使い方のガイダンス

Windows Phone 用 WebBrowser コントロールで説明されているように、Web ビュー コントロールを組み込む理由はさまざまです。

Web ビュー コントロールを使って、リモート Web サーバー、動的に生成されたコード、アプリ パッケージ内のコンテンツ ファイルからリッチ書式の設定された HTML コンテンツを表示できます。またリッチ コンテンツでは、スクリプト コードを含めることができ、さらにスクリプトとアプリのコード間で通信を行うことができます。

Web ビュー コントロールを使ってアプリを開発している場合、「Windows Phone 用の WebBrowser コントロール セキュリティのベスト プラクティス」のセキュリティに関するベスト プラクティスと情報を考慮します。

コントラクト と チャーム

このセクションでは、チャームの設定、デバイス、共有のガイドラインについて説明します。検索コントラクトについて詳しくは、「コントロール」セクションの「検索のガイドライン」をご覧ください。このトピックでは、ファイル ピッカーとファイル ピッカー コントラクトのガイドラインについても説明します。

このセクションの内容

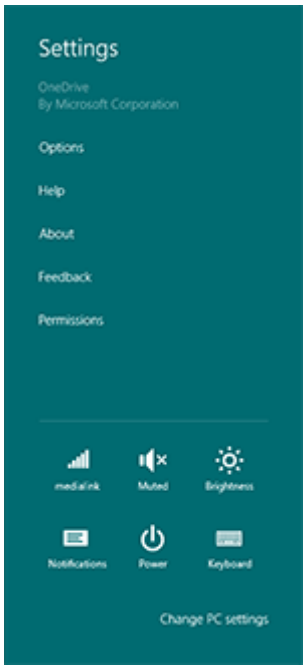
トピック	説明
アプリの設定	このトピックでは、アプリ設定を作成し表示する際のベスト プラクティスについて説明します。
オーディオ認識アプリ	メディア要素では、動画コンテンツとオーディオ コンテンツの再生を管理します。
Windows Phone 用のカメラ レンズ	このトピックでは、Windows Phone でカメラ レンズを使い、リッチ メディアを実装して、一貫性のある優れたレンズ アプリのエクスペリエンスを実現するための設計に関するベスト プラクティスについて説明します。レンズ アプリの開発について詳しくは、「Windows Phone 8 のレンズ」をご覧ください。
カメラの UI	カメラ ダイアログは、内蔵または接続されたカメラから写真やビデオをキャプチャするための、タッチに最適化された全画面表示のエクスペリエンスです。 全画面のダイアログはカメラの UI の表示動作を処理します。
個人データにアクセスするデバイス	マイク、カメラ、位置情報取得機能、テキスト メッセージ サービスは、ユーザーの個人データへのアクセス、またはユーザーへの課金を行うことがあります。そのため、機密性の高いデバイスであると考えられます。 Windows ストア アプリは、こうした機密性の高いデバイスにアクセスを許可するアプリをユーザーが制御できる機能を備えています。このトピックでは、デバイス アクセスを有効または無効にする方法を説明する箇所を設けるように Windows ストア アプリを設計するガイドラインについて説明します。

ファイルピッカー コントラクト	他のアプリにアプリのコンテンツ、保存場所、ファイルの更新へのアクセスを提供するために、ファイル オープン ピッカー コントラクト、ファイル保存ピッカー コントラクト、キャッシュ ファイル アップデーター コントラクトに参加しているアプリのファイル ピッカーをカスタマイズするには、次のガイドラインに従ってください。
ファイルピッカー	アプリは、ファイル ピッカーを使って、ファイルとフォルダーにアクセスし、ファイルを保存できます。
Geofencing	Windows ランタイム アプリの geofencing については、次のベスト プラクティスに従ってください。
位置認識アプリ	このトピックでは、ユーザーの位置にアクセスする必要があるアプリを構築する際のパフォーマンス ガイドラインを説明します。
印刷	Windows ストア アプリでコンテンツの印刷を可能にする場合は、以下のガイドラインに従ってください。
印刷 UI 設計	Windows ストア デバイス アプリのカスタマイズされた印刷 UI を設計するときは、次のガイドラインに従ってください。
近接通信	このトピックでは、近接通信を使ってアプリを接続し、コンテンツを共有する際のベスト プラクティスについて説明します。
コンテンツの共有	このトピックでは、Windows ストア アプリ間でコンテンツを共有するためのベスト プラクティスについて説明します。

アプリ設定のガイドライン

あらゆる Windows ストア アプリに既定で設定ウィンドウが用意されています。設定ウィンドウの上部には、アプリ設定の"エントリ ポイント"の一覧が表示されます。各エントリ ポイントから開いた設定フライアウトには、設定オプション自体が表示されます。このトピックでは、アプリ設定を作成し表示する際のベスト プラクティスについて説明します。

例



注 Windows ストア アプリのあらゆる設定ウィンドウには既定で **[アクセス許可]** と **[評価とレビュー]** という 2 つのエントリ ポイントが追加されます。アプリを Windows ストア経由でインストールしなかった場合 (たとえば、サイドローディングされた企業アプリである場合)、アプリの設定ウィンドウには **[評価とレビュー]** エントリ ポイントが表示されません。設定ウィンドウの下部には、音量、明るさ、電源などのシステム設定が表示されます。

アプリにアプリ設定を含めるかどうか

すべての Windows ストア アプリは設定ウィンドウを持ち、設定ウィンドウには既定で **[アクセス許可]** と **[評価とレビュー]** が表示されます。以下に、設定ウィンドウに追加する設定の例を挙げます。

- アプリ全体の動作に影響するが、頻繁に調整されるわけではないオプションを構成します。たとえば、天気予報アプリで温度の既定の単位として摂氏または華氏を選択する機能や、メール アプリでアカウント設定を変更する機能などです。
- 音楽、効果音、配色テーマなど、ユーザーの設定に基づくオプション。
- プライバシーに関する声明、ヘルプ、アプリのバージョン、著作権情報など、頻繁には必要とされないアプリの情報。
    - 機能でインターネット接続を宣言している場合は、プライバシー ポリシーへのリンク。このポリシーへのリンクを忘れることは、認定が拒否される最も一般的な要因です。 詳しくは、「Windows ストア向けアプリ認定要件」をご覧ください。

アプリの通常のワークフローに含まれるコマンド (お絵かきアプリでのブラシ色の変更など) は設定ウィンドウに含めません。代わりに、よく使うコマンドは上部または下部のアプリ バーに配置します。コマンド配置について詳しくは、「コマンド パターン」をご覧ください。 「アプリ バーのガイドライン」でアプリ バー使用時の推奨事項が説明されています。

推奨と非推奨

一般的な原則

- 設定ウィンドウに、すべてのアプリ設定のエントリ ポイントを作成します。
- 設定はシンプルにします。適切な既定値を定義し、設定の数は最小限にします。
- 必要に応じて、アプリの UI 要素から設定ウィンドウにリンクすることや、特定の設定フライアウトにディープ リンクすることができます。たとえば、下部のアプリ バーにある "[ヘルプ]" ボタンや設定ウィンドウの "[ヘルプ]" エントリ ポイントからヘルプ設定フライアウトにリンクすることができます。
- ユーザーが設定を変更したときは、アプリにすぐに変更が反映されるようにします。設定の変更は即座に適用するか、ユーザーがフライアウトの操作を完了したらすぐに適用します。
- WinJS (**WinJS.UI.SettingsFlyout**) と XAML (**Windows.UI.Xaml.Controls.SettingsFlyout**) で使用できる設定ポップアップ コントロールを使います。これらのコントロールは既定で UI 設計のガイドラインを実装しています (ガイドラインについては「設定 UI の設計」セクションで説明)。

- お絵かきアプリでのブラシ色の変更など、アプリの通常のワークフローに含まれるコマンドはアプリ設定に含めないでください。そのようなコマンドは、アプリ バーまたはキャンバスに配置します。詳しくは、「コマンド パターン」で説明されている概念の概要と「アプリ バーのガイドライン」をご覧ください。
- 設定ウィンドウのエントリ ポイントを使って、直接アクションを実行しないでください。エントリ ポイントは、設定フライアウトを開く必要があります。
- 設定ウィンドウを移動に使わないでください。設定ウィンドウが閉じたときは、設定チャームをクリックしたときと同じアプリ内の位置に戻るようにします。移動には、上部のアプリ バーを使うことをお勧めします。
- 設定ポップアップの各クラスを汎用のコントロールとして使わないでください。これは、設定ウィンドウのエントリ ポイントから起動される設定フライアウトにのみ使います。

エントリ ポイントの追加

エントリ ポイントは、設定ウィンドウの上部に表示されるテキストで、通常、設定フライアウトが開きます。設定フライアウトには、1 つ以上の設定オプションを表示できます。含める設定の一覧ができたら、エントリ ポイントに関する次のガイドラインを考慮してください。

- 1 つのエントリ ポイントに類似したオプションや関連するオプションをまとめます。4 つを超えるエントリ ポイントの追加は避けます。
- アプリのコンテキストに関係なく、同じエントリ ポイントを表示します。いくつかの設定が特定のコンテキストに適合しない場合は、アプリの設定フライアウトでそれらの設定を無効にします。
- エントリ ポイントのラベルは、できればわかりやすい 1 単語にします。たとえば、アカウント関連の設定の場合は、エントリの名前を "アカウント設定" ではなく "アカウント" にします。設定のエントリ ポイントが 1 つだけ必要だが、設定のわかりやすいラベルが思い付かない場合は、"オプション" または "既定" を使います。
- フライアウトではなく直接 Web に移動する場合は、ハイパーリンクとしてスタイルを設定した "ヘルプ (オンライン)" や "Web フォーラム" など、ユーザーにビジュ

アルなツールチップを与えます。Web への複数のリンクは、1 つのエントリ ポイントを使ってフライアウトにまとめることを検討してください。たとえば、"バージョン情報" エントリ ポイントでは、使用条件、プライバシーに関する声明、アプリのサポートへのリンクを含むフライアウトが開くようにします。

- 使用頻度の高い設定にそれぞれ独自のエントリ ポイントを割り当てられるように、使用頻度の低い設定は 1 つのエントリ ポイントにまとめます。主に情報提供のみを行うコンテンツやリンクは、"バージョン情報" エントリ ポイントに配置します。
- [アクセス許可] ウィンドウの機能と重複しないようにします。 このウィンドウは、システムによって制御されており、アプリでは変更できません。

設定 UI の定義

注 WinJS と XAML で使用できる設定ポップアップ コントロールは、ヘッダーと境界線の色に関係する部分を除いて、既定で以下のガイドラインに従っています。これらのコントロール (**WinJS.UI.SettingsFlyout** または **Windows.UI.Xaml.Controls.SettingsFlyout**) を使うと、設定ポップアップが確実に UI のガイドラインに従い、一貫したユーザー エクスペリエンスを提供できます。

- 設定フライアウトは常に設定ウィンドウのエントリ ポイントから起動します。
- メイン アプリのコンテンツの上に表示され、ポップアップの外側をクリックしたり、アプリのサイズを変更すると非表示になる簡易非表示サーフェスを使います。これによって、ユーザーが設定をすばやく変更して、アプリの操作に戻ることができます。
- 設定フライアウトは、チャームおよび設定ウィンドウと同じ側に表示されるようにします。画面のどちらの側に設定チャームを表示するかを決めるには、**SettingsEdgeLocation** プロパティを使います。
- フライアウトは、画面の上端や下端からではなく、設定ウィンドウと同じ側からスライドさせます。
- フライアウトは、向きにかかわらず全画面の高さでなければならず、狭い幅の場合は 346 ピクセル、広い幅の場合は 646 ピクセルにする必要があります。コンテンツに適した幅を選びます。カスタム サイズは作成しません。

- フライアウトのヘッダーには戻るボタン、フライアウトを開いたエントリ ポイントの名前、アプリのアイコンを表示する必要があります。
- ヘッダーの背景色は、タイルの背景色と同じにする必要があります。
- 境界線の色は、ヘッダーと同じ色にしますが、20% 濃くします。
- 設定のコンテンツは白い背景で表示します。

設定フライアウトへの設定の追加

- コンテンツは 1 列で上から下へ表示し、必要に応じてスクロールできるようにします。スクロールの長さは画面の高さの 2 倍までに抑えます。
- アプリ設定では次のコントロールを使うことを検討します。
  - トグル スイッチ: ユーザーが値をオンまたはオフに設定できるようにする場合。
  - ラジオ ボタン : ユーザーが相互排他的な関連するオプション (5 個まで) の中から 1 つの項目を選択できるようにする場合。
  - テキスト入力ボックス: ユーザーがテキストを入力できるようにする場合。ユーザーから取得するテキストの種類 (メール、パスワードなど) に応じた種類のテキスト入力ボックスを使います。
  - ハイパーリンク: アプリ内の別のページや外部 Web サイトに移動する場合。ユーザーがハイパー リンクをクリックすると、現在の設定 UI は閉じられます。
  - ボタン: ユーザーが現在の設定フライアウトを閉じることなく即座に操作を開始できるようにする場合。
- 使用できないコントロールがある場合は、説明用のメッセージを追加します。使用できないコントロールの上に、このメッセージを配置します。
- 設定フライアウトとヘッダーがアニメーション化された後で、コンテンツとコントロールを単一のブロックとしてアニメーション化します。**enterPage** または **EntranceThemeTransition** アニメーションを使って、100 ピクセル左のオフセットでコンテンツをアニメーション化します。
- 必要に応じて、コンテンツの整理と明確化の助けになるように、セクション ヘッダー、段落、ラベルを使います。
- 設定を繰り返し表示する必要がある場合は、UI の階層を追加するか、展開/折りたたみモデルを使います。階層の深さは 2 階層までに抑えます。たとえば、天気予報

アプリの都市別の設定では、都市の一覧を表示し、ユーザーが都市をタップしたときに、新しいフライアウトを開くか、展開して設定オプションを表示できるようにします。

- コントロールや Web コンテンツの読み込みに時間がかかる場合は、進行状況不定コントロールを使ってユーザーに読み込み中であることを示します。詳しくは、「プログレス コントロールのガイドライン」をご覧ください。
- 移動や変更を確定するためのボタンは使いません。別のページに移動するにはハイパーリンクを使います。また、ボタンを使って変更をコミットする代わりに、ユーザーが設定ポップアップを閉じたときにアプリ設定の変更を自動的に保存します。

オーディオ認識アプリの開発のガイドライン

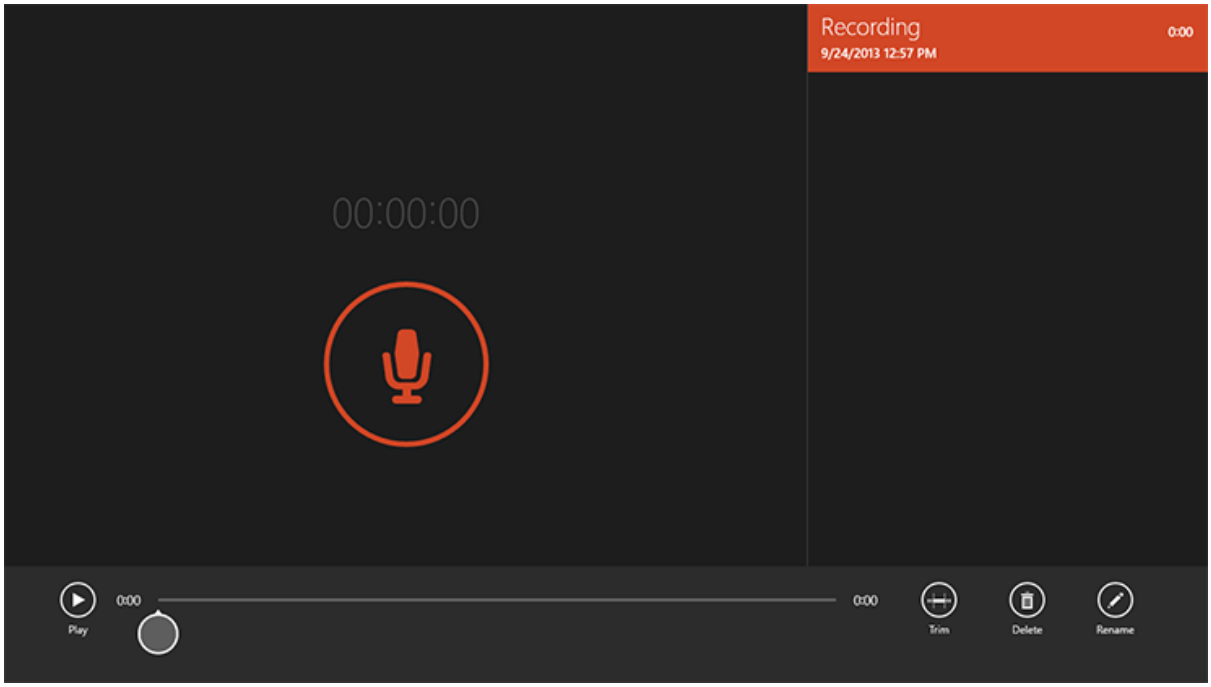
Windows アプリ：メディア要素の既定のトランスポート コントロール

説明

このトピックでは、オーディオ認識アプリケーションの開発のベスト プラクティスを紹介します。具体的には、メディア要素の設計、再生マネージャーの使用、呼び出し制御の管理に関する考慮事項などについて説明します。

Windows Phone: メディア アプリの開発に際しては、いくつかのメディア関連の認定要件を把握しておく必要があります。詳しくは、「Windows Phone 特定のアプリの種類の追加要件」のセクション 6.4 と 6.5 をご覧ください。

例



Windows アプリ：カスタム トランスポート コントロール

推奨と非推奨

メディア要素

- メディア要素を使うと、動画とオーディオを全画面でまたは他のビジュアル効果と一緒にユーザーに表示できます。複数のメディア要素を 1 つの画面で同時に表示できます。また同時に再生しない場合は、どちらを有効にするかをプログラムによって制御できます。
- メディア要素とそのトランスポート コントロールをいつも同じ場所に配置すると、ユーザーがメディアを確実かつシームレスに操作できます。メディア要素は、動画のフレームを示します。トランスポート コントロールは既定では表示されませんが、単純なブール型プロパティでオン/オフを切り替えることができます。全画面ボタンを含むトランスポート コントロールは、メディア要素が最初に表示されたときと、メディア要素をタップしたときに表示され、その後フェード アウトします。

- 独自のカスタム トランスポート コントロールを作る場合、メディア要素が最初に表示されたときにそれらを表示します。その後メディア表示の邪魔にならないようにトランスポート コントロールがフェード アウトしますが、ユーザーがメディア要素をタップするともう一度表示されます。[再生/一時停止] は、タップしたときに行われる動作をビジュアル効果で示したトグル ボタンにする必要があります。このボタンは、メディアの現在の再生/一時停止の状態を表示するために使わないでください。オーディオと動画でスキップ、停止、再生、一時停止、シーク操作を表示するときは、一般にメディア再生で使われているシンボルを使うようにしてください。
- できるだけ遅延時間を短くして応答するようにボタン入力イベント ハンドラーを設計します。これにより、ユーザーは自分のボタン入力をすぐに確認できるようになります。応答の遅延が長いと、ユーザーがボタンを複数回押す原因となり、予想しないアプリの動作を引き起こします。
- メディア ボタンを標準的な方法で使うことができるようにします。これにより、ユーザーが普段と同じような使い心地でメディア ボタンを使えるようになります。
- トラック名とアーティスト名には 127 文字より長い文字列を使わないでください。使った場合、エラーが発生します。このエラーを正しく処理しないと、アプリの機能が停止する場合があります。

Windows ストア アプリのメディア ボタンを追加して構成する方法のチュートリアルについては、「システム メディア トランスポート コントロールの使用方法」をご覧ください。

再生マネージャー

- オーディオをバックグラウンドで再生する必要がある場合のみ、**msAudioCategory**/**MediaElement.AudioCategory** の割り当てを使います。オーディオの再生はバッテリを消費するので、バックグラウンド オーディオが明らかに必要な場合 (たとえば、長時間の視聴用のメディア再生) を除き、オーディオ カテゴリを宣言しないでください。または、「別の」カテゴリを使うこともできます。 そうしなければ、アプリケーションはミュートされてから一時停止します。
- 待機時間が短いオーディオは、特定のアプリケーションにより必要な場合 (マルチトラック レコーダーや待機時間が短い動画のキャプチャなど) にのみ使います。待機時間が短いオーディオは、「通信」オーディオ カテゴリを選ぶと自動的に呼び

出されます。 他のカテゴリの場合、低待機時間設定は既定のまま (OFF) にすることを検討してください。 待機時間が短いバッファーは CPU とバッテリ リソースを非常に多く使いますが、それらは通常、ユーザーがフォーカスするフォアグラウンド アプリケーション用に確保されています。

- バックグラウンド メディアの再生をミュートする場合は、**ForeGroundOnlyMedia** をこのサウンドトラックの **msAudioCategory**/**MediaElement.AudioCategory** として選びます。ユーザーがゲームをしている間に独自のオーディオ サウンドトラックを再生するゲームを開発する場合、ゲームが開始されたときにユーザーがオーディオ トラックを既にバックグラウンドで再生していた場合には、ゲームのサウンドトラックはミュートされます。 ゲームのオーディオ サウンドトラックがゲームの機能に重要だと考える場合は、現在再生中のバックグラウンド オーディオをミュートできます。バックグラウンド メディアと混在する "**SoundEffects**" はどちらの場合も聞こえています。

  "**ForeGroundOnlyMedia**" は、動画開始時にバックグラウンド メディアを停止し、バックグラウンドである間はまったく実行させない必要がある動画アプリに対しても選ぶことができます。

AV ストリームを分類する方法と、Windows ストア アプリでの再生マネージャーの使い方については、「AV ストリームのオーディオ ストリーミング用の分類方法」をご覧ください。

呼び出し制御

次に、既定の Bluetooth 通信デバイスで呼び出し制御を管理する場合のお勧めの方法を示します。

- 呼び出し制御機能を通信アプリケーションに対して予測可能にします。これにより、ユーザーはなじみのあるシームレスな経験をすることになります。 なじみのない方法でアプリで呼び出しボタンを使う場合は、ユーザーにはっきりとわかるようにしてください。
- 呼び出しトークンを注意深く追跡します。デバイスに対し呼び出しを正しく終了し、オーディオ/ビデオ ストリームも終了します。こうすることにより、呼び出しの完了時に、適切な呼び出しトークンにより、「呼び出し終了」通知がデバイスに

送られるようになります。これにより、ユーザーは呼び出しが終了したことをデバイスから示されます。

Windows ストア アプリでの Bluetooth 呼び出しの管理方法のチュートリアルについては、「既定の Bluetooth 通信デバイスで呼び出しを管理する方法」をご覧ください。

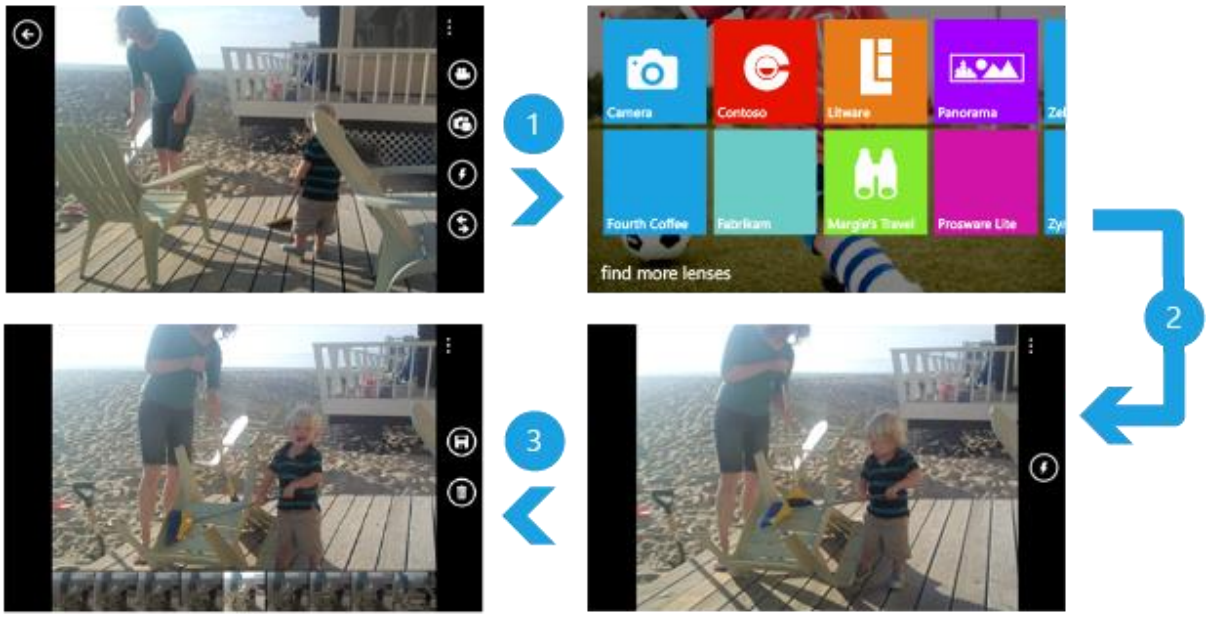
レンズの設計ガイドライン(Windows Phone)

このトピックでは、Windows Phone でカメラ レンズを使い、リッチ メディアを実装して、一貫性のある優れたレンズ アプリのエクスペリエンスを実現するための設計に関するベスト プラクティスについて説明します。レンズ アプリの開発について詳しくは、「Windows Phone 8 のレンズ」をご覧ください。

レンズのユーザー エクスペリエンス ガイドライン

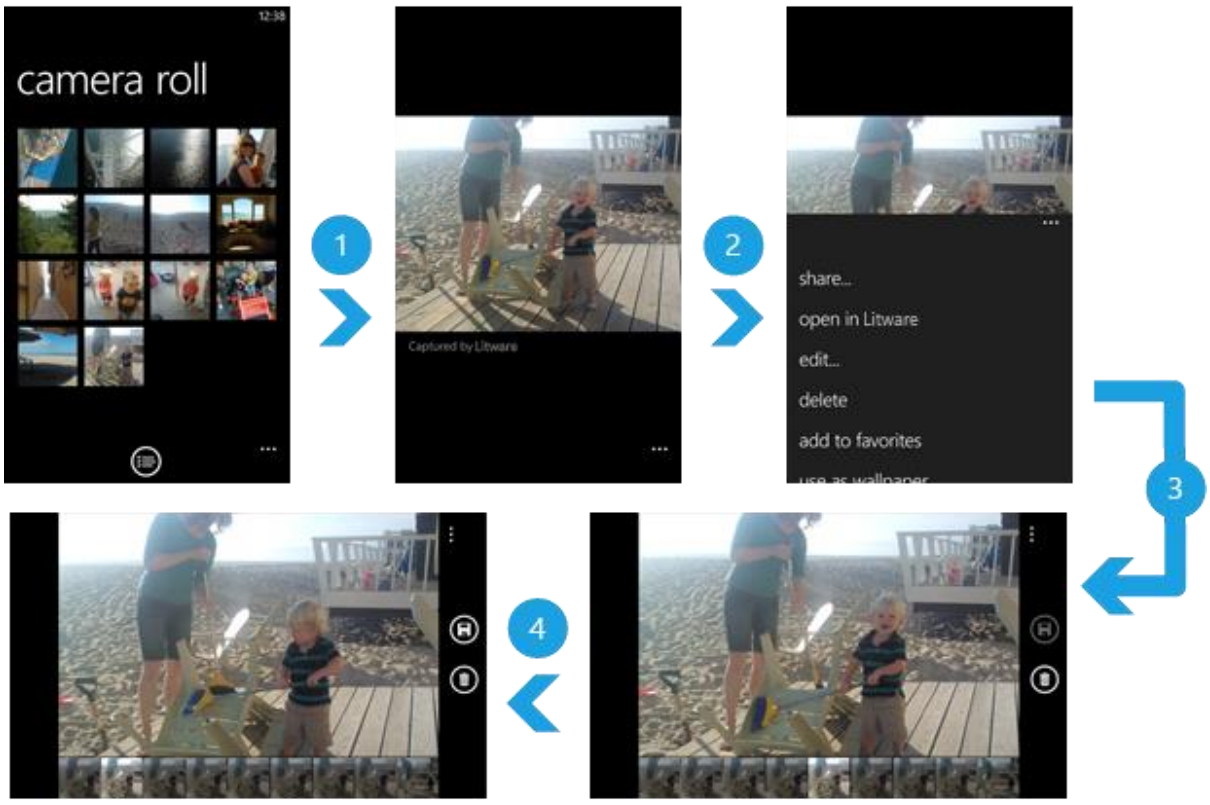
レンズ アプリは、組み込みのカメラを補完するためのもので、カメラと写真表示の両方の機能をそのまま拡張したように感じられます。レンズは、実世界のシナリオを、パノラマやグループ写真など、そのシナリオに特化したアプリに対応付けるのに役立つ手段となります。レンズは、次の 2 つの主要機能を備えることでこれを実現します。

- **キャプチャ**



UI の例: (1) レンズ切り替えボタンをタップし、(2) レンズを選択して瞬間をキャプチャし、(3) 確認して写真をカメラ ロールに保存します。

- **表示と操作：**



UI の例: (1) すべての写真がカメラ ロールに表示され、(2) 写真を作ったアプリでその写真をもう一度開き、(3) 操作または編集し、(4) 新しい写真を保存します。

重要な注意: レンズ アプリを作る場合、次の基本的な点について留意する必要があります。

- レンズはビューファインダーを使用する機能に組み込まれます。
- レンズにより、写真がカメラ ロールに保存されます。
- 強化された表示または編集エクスペリエンスを提供するレンズは、組み込みのフォト ビューアーでそのエクスペリエンスをもう一度提供します。

起動エクスペリエンス

レンズは主にカメラ アプリで、組み込みのカメラ エクスペリエンスのコンテキストで起動されます。デバイスのカメラ エクスペリエンスでは、縦向きと横向きの両方がサポートされますが、ユーザーがデバイスをカメラのように持っているとき (横向き) にレンズ アプリが起動される可能性が最も高いことを理解しておくことが重要です。そのため、起動画面とアプリの既定の向きは横向きに設定することをお勧めします。

レンズはビューファインダーを使用する機能です。つまり、ビューファインダー固有のアプリを起動する場合、ビューファインダーの特性を利用するエクスペリエンスがすぐに実現されます。ただし、ユーザーが資格情報を入力したり、アプリの一部分を使うために法的に同意する必要のあるアプリはこの限りではありません。

組み込みのカメラ エクスペリエンスとのアプリの統合について詳しくは、「Windows Phone 8 のレンズ機能拡張」をご覧ください。

キャプチャ エクスペリエンス

一般的に、レンズのキャプチャ エクスペリエンスには、特別な理由がない限り、組み込みのカメラのユーザー エクスペリエンスとの一貫性を持たせる必要があります。レンズ エクスペリエンスに必要な一貫性を提供するには、次の点を考慮してください。

- ジェスチャ (特に左方向へスワイプ) とエクスペリエンスは、デバイスの向きを基準にして作る必要があります。
- アプリでは、デバイスの向きを基準にして、さらに写真があることを示す左向き矢印アイコンをサポートする必要があります。
- [保存] と [キャプチャ] のストック アニメーションは一貫している必要があります。
- アプリでは、タップによるキャプチャとカメラ ハードウェア ボタンをサポートしている必要があります。
- 半押しによるフォーカスをサポートします。
- 必要な箇所で、フラッシュのアイコン画像と状態を利用できる必要があります。
- 必要な箇所で、基本カメラと同様に、フォーカスを示すかっこが機能する必要があります。

キャプチャの方法

レンズは多くのカメラ アプリで使われるため、利用可能な各種キャプチャ方法と、それぞれの方法に適用される一意の設計ガイダンスを区別することが重要です。

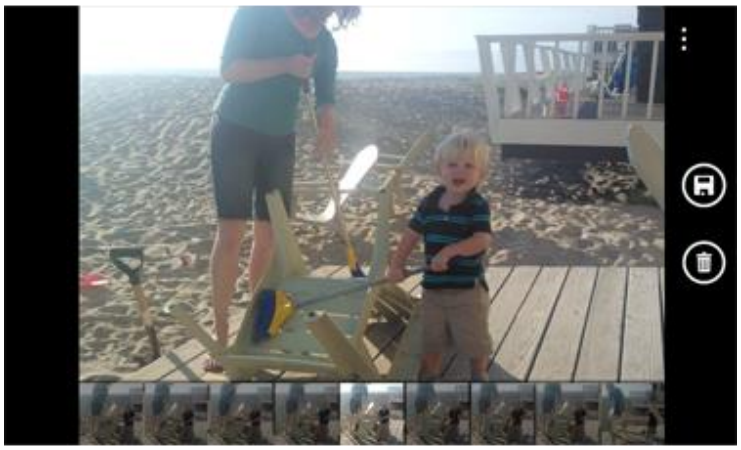
従来のキャプチャ

この種のレンズ アプリでは、写真はカメラ ロールに直接保存され、ユーザーはその後すぐにビューファインダーに戻されます。

従来のキャプチャ アプリ

キャプチャと確認

この種のレンズ アプリでは、キャプチャした画像をカメラ ロールに保存する前に、ユーザーが解析して受け入れる必要があります。



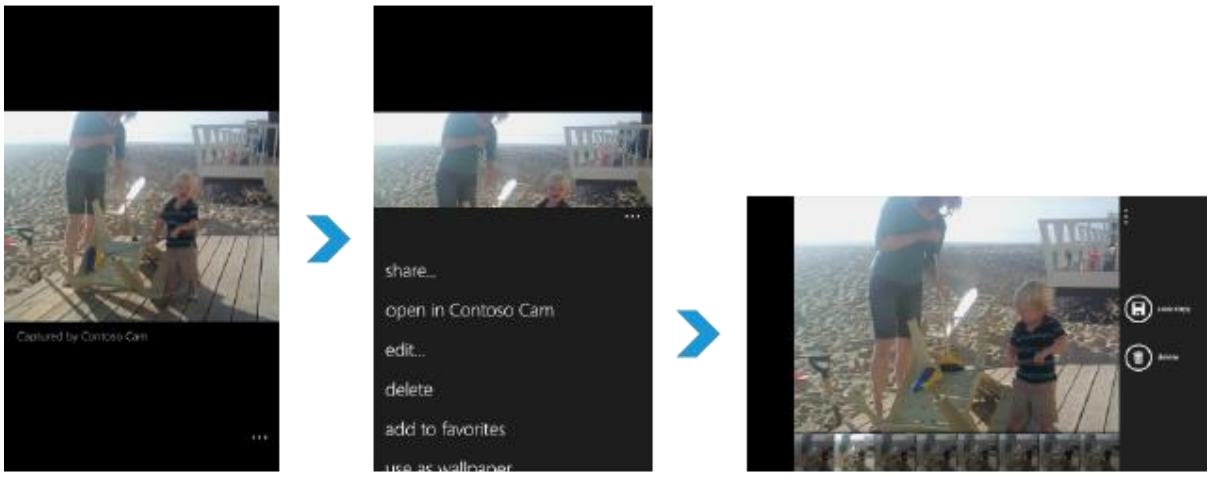
キャプチャと確認を行うアプリ

キャプチャと確認を行うアプリでは、項目の保存を確認、取り消すアニメーションに加え、一貫したアイコン セット ([保存] と [削除]) を使う必要があります。[キャンセル] と [保存] では、いずれもユーザーをビューファインダーに戻す必要があります。これらのアイコンは、Windows Phone SDK に用意されています。

ビューファインダー機能からのリンク バック

多くのレンズ アプリでは、写真は単にユーザーのカメラ ロールに保存されるだけですが、Windows Phone のレンズでは従来よりもはるかに複雑な写真でコンテンツをキャプチャできます。リッチ メディア レンズでは、ユーザーがキャプチャした画像により深くさまざまな形でかかわれるように、ローカル フォルダーまたは Web からデータを組み込みます。

リッチ メディア レンズでは、アプリにリンク バックされる写真を保存できます。カメラ ロールに保存された写真は、カメラ ロールの他の写真と同様に共有、編集できます。組み込みのフォト ビューアーでリッチ メディア項目を表す写真を表示する場合、ユーザーはその項目に関連付けられているリッチ メディア エクスペリエンスにリンク バックできます。



オープン リンクからは、選んだ項目を表示または編集するようにカスタマイズされた機能を起動する必要があります。これは、アプリの一般的な起動ポイントと考えないでください。[コピーの保存] は、ユーザーが画像を変更した場合のみ有効になるようにします。リッチ メディア レンズを組み込みのフォト ビューアーと統合する方法について詳しくは、「Windows Phone 8 のリッチ メディア機能拡張」をご覧ください。

リッチ メディア アプリでは、カメラ ロールに保存した画像があるとは想定しないでください。ユーザーはカメラ ロールに保存した項目を削除できます。したがって、カメラ ロールに保存された対応する画像がない場合でも、リッチ メディア レンズでもう一度エクスペリエンスを作成できるようにしてください。

ユーザーは、カメラ ロールに保存された項目を共有または編集できるので、ブランド要素の使用は避けてください。不要で目障りな表示に対処することなく、ユーザーが画像を共有できるようにします。

キャプチャした写真と関連付けられたバッキング データは、アプリのローカル フォルダーに累積されます。リッチ メディア アプリでは、カメラ ロールから画像を削除することはできませんが、ローカル フォルダーからデータを消去することはできます。これらのアプリでは、アプリでキャプチャした任意の写真に移動する機能が用意されているため、ユーザーは、その写真と関連付けられているバッキング データを削除できます。リッチ メディア レンズでアプリから項目の新しいコピーが作られる場合、このアクションは [保存] ではなく [コピーの保存] で行われる必要があります。

次に、リッチ メディア エクスペリエンス内の移動に関するヒントをいくつか示します。

- 表示または編集エクスペリエンスを開始する場合は、[戻る] でカメラ ロールに戻る必要があります。
- 編集エクスペリエンスを開始する際、[コピーの保存] では、確認した変更を表示するためにユーザーをアプリ内に留める必要があります。[削除] では、画像に関連付けられているバッキング データを削除します。

アプリでリッチ メディア エクスペリエンスが提供されていない場合は、アプリの WMAppManifest.xml ファイルでリッチ メディア拡張機能を宣言しないでください。

注 リッチ メディア項目が保存されないアプリには、[削除] オプションを提供しないでください。代わりに、現在のセッションでキャプチャされた項目を表示します。

デザインに関するその他の考慮事項

レンズは強力なアプリですが、その機能は限定的です。ユーザーのカメラ ロールから写真を削除したり、ユーザーがインストールしたその他のレンズを列挙したり、レンズ アプリで組み込みの編集エクスペリエンスを開始したりすることはできません。これらの制限は、アプリのユーザーの個人情報とデータを保護することを目的としています。レンズでは、電話に組み込まれているフォト ビューアーのすべての機能を模倣しないでください。

レンズ ピッカー用のアイコンの準備

レンズ ピッカーには、アプリ自体を示すアイコンとは異なる解像度のアイコンが必要です。アプリでは、Assets フォルダーに、電話で使用可能な解像度ごとに 3 つのアイコンを用意する必要があります。各アイコンの名前と解像度に関する説明を次の表に示します。

解像度ごとにアイコンを作る方法について詳しくは、Windows Phone 8 のアイコン テンプレートをダウンロードしてください。

電話の解像度	アイコン サイズ(ピクセル)	フォルダー	ファイル名
WVGA	173 x 173	Assets	Lens.Screen-WVGA.png
HD720p	259 x 259	Assets	Lens.Screen-720p.png
WXGA	277 x 277	Assets	Lens.Screen-WXGA.png

電話の解像度について詳しくは、「Windows Phone 8 の複数解像度アプリ」をご覧ください。

推奨事項のまとめ

すべてのレンズは、ビューファインダーを使用する機能に組み込まれ、写真をカメラ ロールに保存します。その他の留意点を以下にまとめます。

起動エクスペリエンス:

- レンズのスプラッシュ スクリーンは横向きに表示されます。
- レンズ アイコンでは、WVGA、HD720p、WXGA の解像度がサポートされます。

キャプチャ エクスペリエンス:

- カメラの既定のユーザー エクスペリエンスと一貫性があるようにします。
  - ジェスチャ サポート: プレビューするための左方向へのスワイプ。
  - 横方向と縦方向をサポートします。
  - ボタンの動作:
    - 半押し。
    - ハードウェアのキャプチャ。
    - タッチによるキャプチャ (フォーカスあり)。
  - 必要な場所におけるフラッシュのアイコンとオン、オフ、自動の状態、前面カメラ。
  - フォーカスのかっこ。
- カメラ ロールにはキャプチャごとに 1 つの写真が保存されます。
- キャプチャで複数の JPG 画像が作成される場合、アプリのローカル フォルダーに追加のバッキング データが保存されるようにします。

キャプチャと確認を行うアプリ:

- [保存]、[コピーの保存]、[削除] には一貫したアイコンのセットを使います。
- [削除] と [保存] ではいずれもビューファインダーに戻る必要があります。

リッチ メディア レンズ:

- アプリで写真を編集したり、後で表示したりするために追加のデータが保存される場合、リッチ メディア機能の実装を検討する必要があります。
- オープン リンクでは、選んだ項目を表示または操作するようにカスタマイズされた機能にユーザーを移動します。
- (ユーザーが画像を削除した場合があるため) カメラ ロールを開く前にカメラ ロール内に画像があるかどうかを確認し、画像がない場合は適切に処理します。
- リッチ メディア エクスペリエンスを提供するアプリでは、ユーザーがカメラ ロールの項目からリンクしたデータがアプリで削除されている場合にも対応できる必要があります。
- リッチ メディア レンズ アプリでは、アプリでキャプチャされたコンテンツを、カメラ ロールではなくローカル フォルダーのバッキング データに基づいて列挙する必要があります。

- リッチ メディア レンズ アプリでは、ユーザーがデバイスからバッキング データを削除できる必要があります。
- 編集エクスペリエンスを開始する場合、保存機能は [コピーの保存] と呼ばれます。アプリのユーザーには、確認した変更が表示されるようにします。
- オープン リンクからの移動:
    - 表示または編集エクスペリエンスを開始する場合、[戻る] を押すと、カメラ ロールに戻る必要があります。

アプリでリッチ メディア エクスペリエンスが提供されない場合:

- リッチ メディアが保存されないアプリには、[削除] オプションを提供しないでください。代わりに、現在のセッションでキャプチャされた項目を表示します。
- アプリでリッチ メディアを使わない場合は、アプリの WMAppManifest.xml ファイルではリッチ メディア拡張機能を宣言しないでください。

カメラの UI のガイドライン

カメラ ダイアログは、内蔵または接続されたカメラから写真やビデオをキャプチャするための、タッチに最適化された全画面表示のエクスペリエンスです。 全画面のダイアログはカメラの UI の表示動作を処理します。このダイアログにより、**Windows.Media.Capture.CameraCaptureUI.captureFileAsync** API への 1 つのメソッド呼び出しを使って、写真やビデオをキャプチャできます。キャプチャのエクスペリエンスの一部として、呼び出し元のアプリケーションに戻す前に、ユーザーはキャプチャした写真やビデオをトリミングできます。さらに、写真やビデオをキャプチャする前に、明るさ、コントラスト、露出など、カメラの一部の設定を調整することもできます。 カメラ ダイアログは、ライブ フォトやライブ ビデオのキャプチャを目的としています。

推奨と非推奨

- アプリケーションがライブ フォトまたはライブ ビデオを必要とする場合に、カメラの UI を使います。たとえば、プロファイル画像を必要とするアプリケーションで、ライブ キャプチャを開始するカメラ ダイアログを表示することで、プロファイル画像が最新であることを確認できます。
- リアルタイムのフィードバックを得る、またはキャプチャ中の画像を制御する場合は、カメラの UI を使わないでください。 たとえば、バーコード リーダー アプリで、カメラを使ってバーコードをスキャンする際に、ユーザーにリアルタイムのフィードバックを提供し、それによりユーザーはバーコードが読み取り可能かどうかを知ることができます。この場合カメラ ダイアログは、キャプチャされたビデオ ストリームを直接制御できないため、適切なオプションではありません。代わりに **MediaCapture** API を使ってください。
- ユーザー インターフェイスにカスタムのコントロールを追加する必要がある場合、カメラ ダイアログは使わないでください。 カメラ ダイアログで提供されるものを超えた UI のカスタマイズを追加する必要がある場合は、代わりに **MediaCapture** を使ってください。
- アプリケーションで提供される場合には、カメラ ダイアログでトリミングをオンにしないでください。 ビデオや写真の編集アプリケーションの場合、または写真やビデオの編集機能を提供する場合、トリミングをオフにしてカメラ ダイアログを使う必要があります。そうすると、アプリケーションのトリミング機能はカメラ ダイアログが提供する機能と重複しません。

個人データにアクセスするデバイスのガイドライン

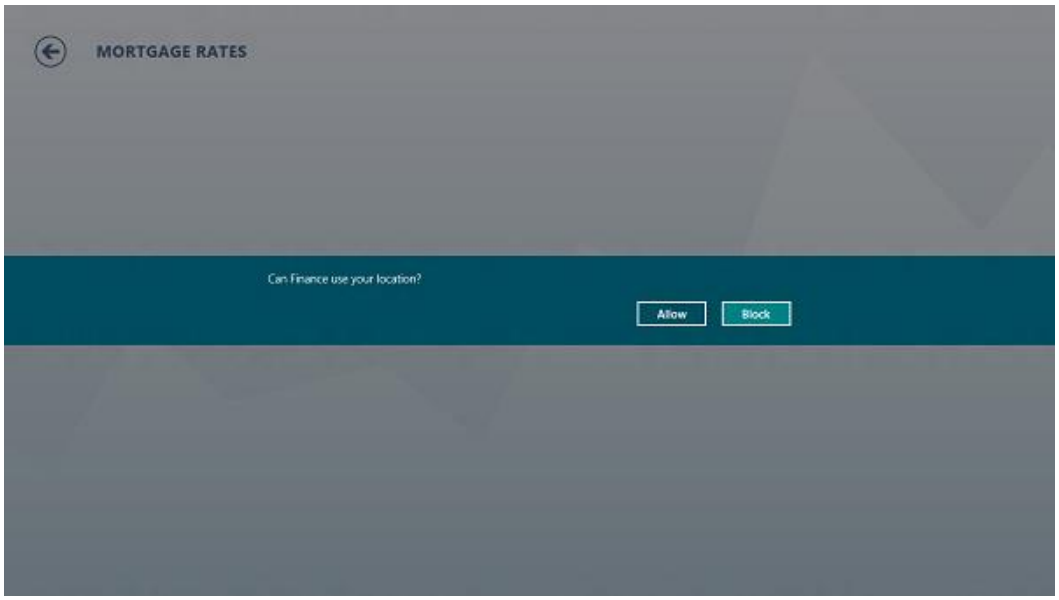
マイク、カメラ、位置情報取得機能、テキスト メッセージ サービスは、ユーザーの個人データへのアクセス、またはユーザーへの課金を行うことがあります。そのため、機密性の高いデバイスであると考えられます。 Windows ストア アプリは、こうした機密性の高いデバイスにアクセスを許可するアプリをユーザーが制御できる機能を備えています。このトピックでは、デバイス アクセスを有効または無効にする方法を説明する箇所を設けるように Windows ストア アプリを設計するガイドラインについて説明します。

例

Windows ストア アプリが機密性の高いデバイスを使うためのアクセス許可は、アプリごと、ユーザーごとに制御します。ユーザーは、同意のプロンプトを使うか設定チャームを使ってアクセス許可を制御します。次の図は、同意のプロンプトに関連付けられた UI と設定ウィンドウの **[アクセス許可]** セクションを示しています。

同意のプロンプト

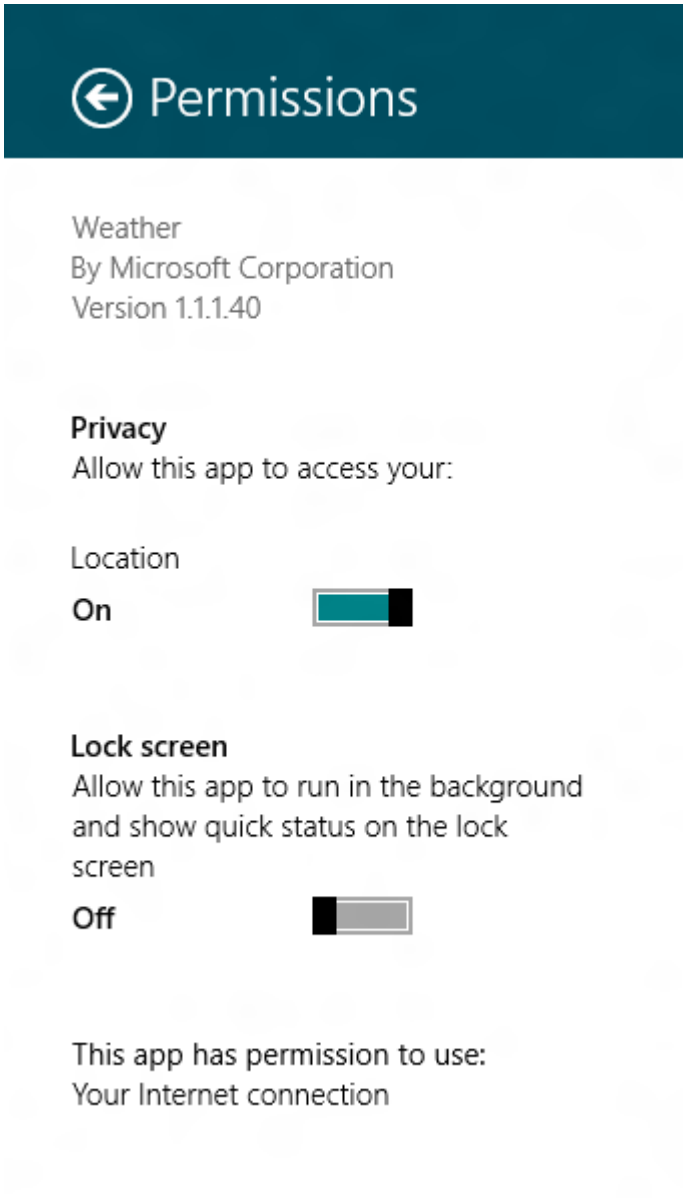
次の図、Windows ストア アプリに表示された同意のプロンプトを示しています。このプロンプトは、アプリが機密性の高いデバイスに初めてアクセスを試みるときに表示されます。ユーザーは、デバイス機能へのアプリのアクセスをブロックするか許可するかを選ぶことができます。Windows ではその応答が記憶されるので、同じアプリについて確認のメッセージがもう一度表示されることはありません。



設定チャーム

各アプリから機密性の高いデバイスへのアクセスは、設定チャームを使って制御することもできます。設定チャームをタップし、設定フライアウトを開きます。設定チャームは、以下の画像に示すように、アプリの右側にあります。

あらゆる Windows ストア アプリに既定で [アクセス許可] フライアウトが用意されています。ユーザーが設定ウィンドウで [アクセス許可] をクリックすると、[アクセス許可] フライアウトが表示され、アプリからの機密性の高いデバイス機能へのアクセスを有効または無効にできます。天気予報アプリの [アクセス許可] フライアウトを次に示します。



推奨と非推奨

- ユーザーに同意のプロンプトを表示するために、デバイスを使い始めるための最初の呼び出しはメイン UI スレッドで行う必要があります。同意のプロンプトを表示できない場合、ユーザーはアプリにデバイスへのアクセスを付与できません。次のことを確認してください。
  - デバイスの初めての使用でバックグラウンド タスクを使わないでください。
  - JavaScript を使うアプリでは、デバイスにアクセスするオブジェクトを初めて使う際には、アプリのアクティブ化ハンドラーでそのオブジェクトを使わないでください。

    - C# または C++ と XAML を使うアプリでは、デバイスにアクセスするオブジェクトを初めて使う際には、通常は App.xaml.cs ではなく MainPage.xaml.cs でそのオブジェクトを使ってください。
- 機密性の高いデバイスの使用がアプリにとって必須ではない場合は、ユーザーがそれらのタスクを実行するまでデバイスにアクセスしないでください。たとえばソーシャル ネットワーキング アプリに、"位置情報を使ってチェックイン" や "プロフィール写真を撮る" というボタンがある場合、このアプリでは、ユーザーが対応するボタンをクリックするまで位置情報やカメラにアクセスしないようにします。
- アプリの主な機能のためにデバイスへのアクセスが必要な場合は、アプリの起動時にデバイスにアクセスできます。たとえば、ライブ動画をキャプチャするアプリでは、その主な目的のためにカメラが必要です。この場合、デバイスへのアクセスを即座に要求することが適切です。
- 設定ウィンドウの **[アクセス許可]** ページをプログラムを使って起動しないでください。

デバイスへのアクセスがオフになっている場合の対処方法

デバイスへのアクセスが無効になる可能性があるのは、同意を求めるメッセージに対してユーザーが同意を拒否した場合、ユーザーが設定ウィンドウの [アクセス許可] ポップアップでアクセスをブロックした場合、またはデバイスがシステムに存在しない場合の 3 つ場合です。アプリが機密性の高いデバイスにアクセスできない場合は、次の推奨事項に従ってください。

- ユーザーがデバイスを使おうとしたときは、デバイス機能を利用できないことをユーザーに通知します。ユーザーが機能を使うことができないことを認識する必要があります。
- デバイスが無効であることを示す通知は、ユーザーに明確に表示されるようにします。
- アプリの主な機能に必要な機能ではない場合、フライアウトまたはインライン メッセージを使ってユーザーに通知します。
- アプリが無効になっているデバイス機能にアクセスしようとしたときに発生する API からのエラーを処理します。アプリがデバイスにアクセスできないことを示す

エラーを返す場合があるメソッド呼び出しについて詳しくは、このトピックの「リファレンス」をご覧ください。

- デバイス機能が無効になっていることを通知し、設定チャームを使ってもう一度有効にしてからアプリで機能の使用を再試行する方法について説明するメッセージをユーザーに表示します。デバイスが無効であることをユーザーに通知する方法の例については、「その他の使い方のガイダンス」をご覧ください。
- ユーザーがデバイスをもう一度有効にした場合にデバイスへのアクセスを再開するための UI を用意します。この UI を使ってデバイスにアクセスするオブジェクトを再インスタンス化または再初期化します。たとえば、地図アプリに現在の位置情報を更新するためのボタンがあるとします。このボタンは新しい **Geolocator** オブジェクトをインスタンス化する必要があります。
- デバイス機能を利用できないことをユーザーに通知するために通知を使わないでください。
- ユーザーがまだ要求していないデバイス機能についてエラー メッセージを表示しないでください。たとえば、ソーシャル ネットワーキング サイトに、ユーザーがメッセージを投稿するときに位置情報を含めるオプションがあるが、ユーザーが位置情報を共有するように選んでいない場合は、メッセージの投稿時にエラー メッセージを表示しないでください。

その他の使い方のガイダンス

エラー メッセージのサンプル

デバイスが無効になっている理由	エラー メッセージ形式のサンプル
ユーザーが同意のプロンプトまたは設定チャームを使ってアクセスをブロックした。	"<device capability> は現在無効になっています。<device capability> の設定を変更するには、設定チャームを開き、[アクセス許可] をタップします。次に、<enable action> を行い、<device capability> の使用を再開してください。" • <device capability> は Web カメラ、マイク、位置情報、またはテキスト メッセージングに置き換えます。 • <enable action> は、機能へのアクセスを再初期化するためにユーザーが UI で実行する必要がある操作 (ボタンのクリックなど) に置き換えます。
デバイス機能がシステムに存在しない。	"必要な <device capability> がシステムに存在しません。"

無効になっているデバイス機能に関するメッセージの表示に使う UI は、そのデバイス機能がアプリにとって必須かどうかによって異なります。次の例は、デバイス機能のメッセージの表示方法を示しています。

デバイスが必須ではない場合のフライアウトまたはインライン テキストの表示

機密性の高いデバイスがアプリにとって必須ではない場合は、呼び出し時にフライアウトでメッセージを表示するか、控えめなインライン テキストでメッセージを表示します。

たとえば、地図アプリに現在の位置情報を表示するためのボタンがあり、必要なデバイス機能が無効になっているときにユーザーがそのボタンをクリックした場合は、アプリではボタンの近くにフライアウトでエラー メッセージを表示するか、インライン テキストでエラー メッセージを表示する必要があります。

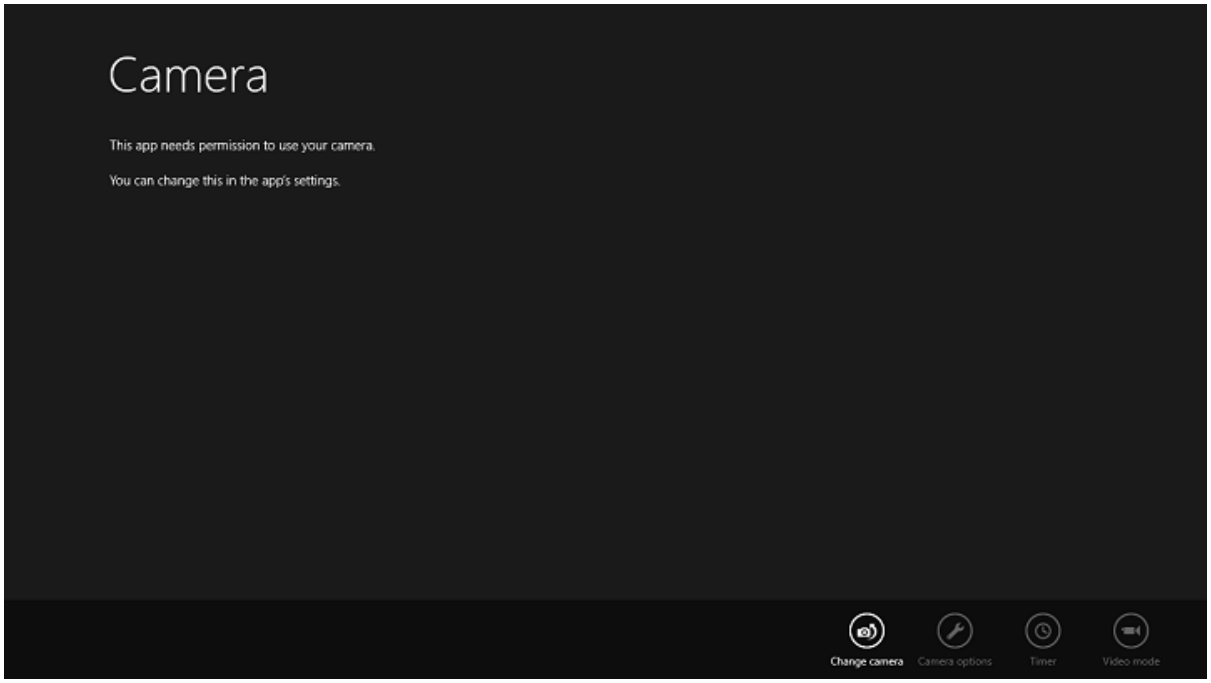
次のスクリーン ショットは、フライアウトでメッセージを表示するアプリを示しています。メッセージの内容は、"位置が見つかりません。[設定] でアクセス許可を変更して、地

図が位置情報にアクセスできるようにしてください。その後、アプリを再起動します。" というものです。

デバイス機能が必須である場合のダイアログの表示

アプリの主な機能のためにデバイスへのアクセスが必要な場合は、**MessageDialog** を使ってエラー メッセージを表示します。

次のスクリーン ショットは、カメラ アプリがその主な機能のために Web カメラとマイクの機能へのアクセスを必要とするようすを示しています。そのため、カメラ機能を有効にするように指示するメッセージを表示します。



リファレンス

次の表に、機密性の高いデバイス機能を無効にしたりもう一度有効にしたりする場合の API の情報を示します。

機能	有効化と無効化の情報
位置情報	**GetGeopositionAsync** メソッドまたは **PositionChanged** イベントのイベント ハンドラーによって同意のプロンプトがトリガーされます。 アプリでユーザーによる位置情報の無効化と再有効化を処理する方法について詳しくは、「位置認識アプリのガイドライン」をご覧ください。

Web カメラまたはマイク	**Windows.Media.Capture.CameraCaptureUI.CaptureFileAsync** を使ってカメラから写真またはビデオをキャプチャするアプリでは、次の点に注意してください。 • **Windows.Media.Capture.CameraCaptureUI.CaptureFileAsync** を呼び出すと、アプリを初めて実行したときに同意のプロンプトがトリガーされます。Web カメラ機能がオフになっている場合、**Windows.Media.Capture.CameraCaptureUI.CaptureFileAsync** はエラーを返しません。代わりに、カメラ キャプチャ UI は、Web カメラ機能がオフになっていることを示すメッセージを表示します。 **Windows.Media.Capture.MediaCapture** を使ってオーディオ、ビデオ、または写真をプレビューまたはキャプチャするアプリでは、次の問題に対処する必要があります。 • **MediaCapture** の非同期メソッドのエラー ハンドラーは、ユーザーがアクセスを許可しなかった場合は E_ACCESSDENIED エラーを、アクセス許可が無効になっている場合は HRESULT_FROM_WIN32(ERROR_FILE_HANDLE_REVOKED) を受け取ります。 • ユーザーが Web カメラへのアクセスを無効にした後でもう一度有効にした場合は、**InitializeAsync** をもう一度呼び出してカメラにアクセスします。たとえば、HRESULT_FROM_WIN32(ERROR_FILE_HANDLE_REVOKED) エラーのエラー ハンドラーが、設定チャームを使って Web カメラをもう一度有効にしてからビデオのプレビューを再開するボタンをタップするようにユーザーに指示するとします。ボタンの背後にあるコードは、他の呼び出しを行う前に **InitializeAsync** を呼び出す必要があります。 **IAudioClient2** インターフェイスを使うアプリでは、**ActivateAudioInterfaceAsync** の呼び出しによって同意のプロンプトがトリガーされることに注意してください。

ファイル ピッカー コントラクトのガイドライン

他のアプリにアプリのコンテンツ、保存場所、ファイルの更新へのアクセスを提供するために、ファイル オープン ピッカー コントラクト、ファイル保存ピッカー コントラクト、キャッシュ ファイル アップデーター コントラクトに参加しているアプリのファイル ピッカーをカスタマイズするには、次のガイドラインに従ってください。

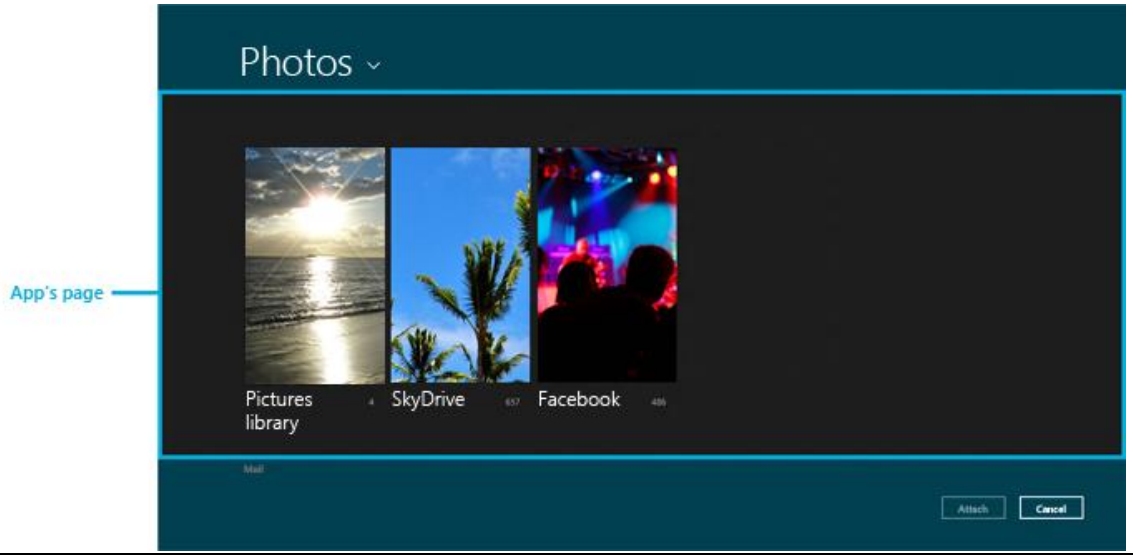
推奨と非推奨

- ファイルを提供するファイル オープン ピッカー コントラクトと統合すると、ファイル ピッカーを通じてアプリのコンテンツにユーザーと他のアプリがアクセスできるようになります。
- 保存場所を提供するファイル保存ピッカー コントラクトと統合すると、ファイル ピッカーを通じてユーザーと他のアプリに保存場所を提供できます。
  アプリで保存場所を提供する場合は、ファイル オープン ピッカー コントラクトと統合して、アプリのコンテンツへのアクセスも提供する必要があります。
- ファイルのリアルタイム更新を提供するキャッシュ ファイル アップデーター コントラクトと統合すると、アプリのリポジトリにあるファイルの更新を実行したり、リポジトリにあるローカル バージョンのファイルに更新を提供したりできます。
  ユーザー側からは、これによってアプリのリポジトリにあるリモート ファイルをまるでローカルにあるかのように操作することができます。たとえば、ユーザーはテキスト エディター アプリを使ってファイルを編集し、Microsoft OneDrive のリポジトリにあるそのファイルのバージョンを更新することができます。
  アプリでファイルを更新する場合は、ファイル保存ピッカー コントラクトとファイル オープン ピッカー コントラクトと統合して、保存場所とファイルへのアクセスも提供する必要があります。

その他の使い方のガイダンス

アプリでファイル ピッカーを使って、ファイル、保存場所、ファイルの更新を提供する場合、ユーザーにファイル (または他の UI) を表示するためのページを設計することが必要になります。このページは、ファイル ピッカーの中央の領域に表示されます。このページと

ファイル ピッカーについて詳しくは、「ファイル ピッカー コントラクトとの統合」をご覧ください。



このスクリーン ショットは、アプリのページ (ファイル ピッカーのページ) が読み込まれる場所がわかるように、ファイル ピッカー ウィンドウの中央の領域を強調表示してラベルを付けたものです。

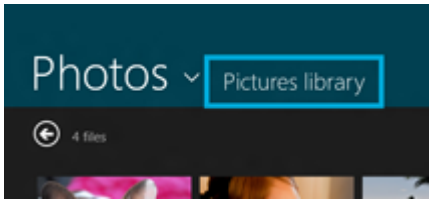
- **すべてのサイズのウィンドウに対応するようにファイル ピッカーのページを設計する**

  Windows 8.1 では、320 ピクセルほどの狭いウィンドウにファイル ピッカーを表示します。狭いディスプレイを最大限に活用できるように、ファイル ピッカーのページの幅が 500 ピクセル未満のウィンドウに読み込まれたときに、ページの左余白を 20 ピクセルに減らし、また垂直スクロールを使うことを検討してください。Microsoft Visual Studio のファイル ピッカー項目テンプレートは、幅が 500 ピクセル未満のウィンドウでは垂直スクロールをサポートし、それよりも大きいウィンドウでは水平スクロールをサポートしていることに注意してください。
- **アプリでファイルを表示するために既に使っているページを基に、ファイル ピッカーで表示するページ (ファイル ピッカーのページ) を設計する**

  ユーザーがファイル ピッカーを使って選ぶファイルをアプリで提供する場合、ユーザーがファイルを表示するためのページがアプリに既に用意されている必要があります。ファイル ピッカーのページを設計するときは、お使いのこのファイル ビュー ページとの一貫性が維持されたページにすることをお勧めします。この 2 つ

のページに一貫性を持たせると、ユーザーはファイル ピッカーでファイルを表示するときに、見慣れたページで快適に操作できるようになります。
ファイル ピッカーのページでの快適な操作をさらに確実にするには、ファイル ピッカーのページで、アプリでお使いのファイル ビュー ページと同じ (または類似した) ナビゲーション UI とエラー報告を使います。特にナビゲーションについては、ファイル ピッカーのページと、お使いのファイル ビュー ページでほぼ同じコマンドと場所を利用できることをユーザーは求めています。

- **ユーザーの現在の作業に関係するファイル ピッカーのページを設計する**
  ファイル ピッカーのページでは、直接関係のない UI を削除して、ファイルの選択、保存、更新など、ユーザーの現在の作業に焦点を合わせた UI を用意します。これにより、ファイル ピッカーを使った操作で、ユーザーが使っていた元のアプリ (呼び出し元アプリまたは呼び出し元) にすばやく戻れるようになります。
  たとえば、アプリが提供するファイルにアクセスするためにファイル ピッカーを使う場合は、選ぶことができない複雑な (詳しい) ナビゲーション、検索、情報をサポートする UI を削除します。
  ユーザーがファイルの使用、変更、管理などの他の作業を行うことができるようにする場合は、その作業用のコントロールまたは他の UI をメイン アプリに追加します。コントロールを追加する方法については、「コントロールとコンテンツの追加」をご覧ください。
- **ファイル ピッカーのタイトルをユーザーの現在の場所の名前に設定する**
  これにより、ユーザーがファイル ピッカーからアプリを使うときに、現在の場所を確認しやすくなります。タイトル (このスクリーン ショットで強調表示されている部分) は、ファイル ピッカーのレターボックスのトップ バーに表示されます。
  このスクリーン ショットでは、タイトルが "ピクチャ ライブラリ" になっているため、ユーザーは自分がシステムのどこにいるのかわかります。このタイトルは、ユーザーが別の場所に移動するたびに更新する必要があります。



- **アプリからアクセスできるすべてのファイルの場所にファイル ピッカーのページからアクセスできるようにする**
  アプリで通常アクセスできる場所にあるファイルには、ファイル ピッカーのページからもアクセスできるようにする必要があります。また、アプリにファイル ピッカーのページが複数ある場合は、アクセスできる場所がすべてのページで一貫している必要があります。そうすると、ユーザーがファイルや場所に予想どおりにアクセスできるようになります。
- **Microsoft Visual Studio の UI テンプレートと UI コントロールを使う**
  Visual Studio には、Windows ストア アプリでファイル ピッカーのビューを作成するときに使うことができる組み込みのテンプレートが用意されています。
- **ユーザーがファイル ピッカーからアプリを起動したときに、サインインとセットアップの対話操作をシンプルに保つ**
  サインインやセットアップのタスクがシンプルな (1 回の操作で実行できる) 場合は、コンテキストを変更する必要がないように、ユーザーがファイル ピッカーからタスクを完了できるようにしてください。共有チャームから複数の対話操作を完了するようユーザーに要求することは避けてください。代わりに、複雑な対話操作はアプリを直接開いて完了するようユーザーに伝えてください。メイン アプリを使って複雑な対話操作を完了する場合は、そのようなタスクをわかりやすく効果的に整理するためのスペースがあることを確認します。
  サインインまたはサインアップの対話操作で Web 認証ブローカーを使うことを計画している場合は、「Web 認証ブローカー」をご覧ください。

追加の UX ガイドライン: ファイル オープン ピッカー コントラクト

- **アプリ固有の適切な方法でファイル ピッカーのページにファイルを表示する**
  アプリ固有の方法でファイルを整理および表示して、ユーザーに合った使いやすいページになるようにします。ただし、その方法は、アプリでファイルの表示に使われているビューと一貫している必要があります。
- **ファイル ピッカーのページには、Windows や他のアプリからアクセスできないファイルを表示する**

他のアプリや Windows からはアクセスできない場所 (アプリの保存フォルダー、リモート サーバーなど) にあるファイルにアクセスできるようにして、Windows や他のアプリとの差別化を実現します。

- **ファイル ピッカーのページでは、呼び出し元アプリの選択モードに応じた UI を設計する**

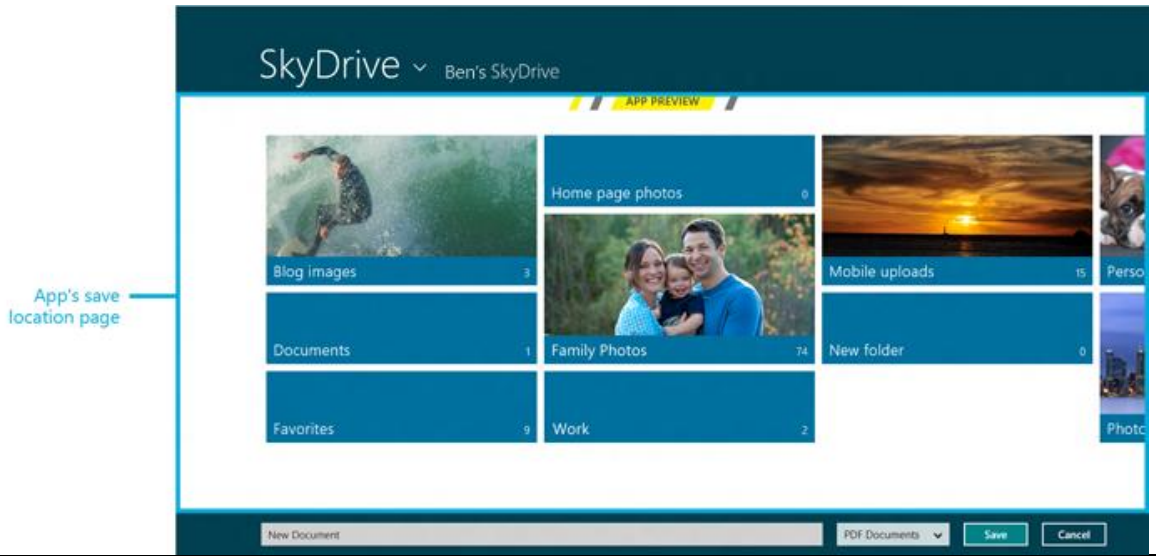
  アプリでファイル ピッカーを呼び出してファイルにアクセスする場合、その呼び出し元のアプリによって、ユーザーが単一の項目を選べるか複数の項目を選べるかが指定されます。選択対象ファイルを選択モードごとに異なる方法で適切に示すようにアプリのページを設計することをお勧めします。たとえば、ユーザーがアプリで用意されているファイルからプロフィール画像を選ぶ場合 (単一項目の選択)、どの写真を選ぶか決めるまでに複数の写真をタップまたはクリックする可能性があります。この場合、アプリの UI で一度に 1 項目しか選べないようにします。一方、ユーザーが友人と共有する複数のファイルを選ぶ場合は (複数項目の選択)、アプリの UI で一度に複数の項目を選べるようにします。

- **Web カメラ アプリ、写真アプリ、カメラ アプリの場合、ファイル ピッカーのページでは写真撮影に関係する UI を設計する**

  ファイル ピッカーのページではアプリの UI をシンプルにして、ユーザーが使っていた元のアプリ (呼び出し元アプリまたは呼び出し元) に確実に戻れるようにします。ファイル ピッカーのページに用意するコントロールは、ユーザーが写真を撮ったり、少数の前処理効果 (フラッシュの切り替えやズームなど) を適用したりするためのコントロールに限定します。

  ファイル ピッカーからアプリ バーにアクセスすることはできないため、使用可能なすべてのコントロールをファイル ピッカーのページに表示する必要があります。ファイル ピッカーのページに表示するコントロールは、アプリ バーのコントロールと同様の構成にし、アプリ バーでコントロールが表示される場所のできるだけ近くに (ページの上部または下部)、ファイル ピッカーのページのコントロールを配置することをお勧めします。

追加の UX ガイドライン: ファイル保存ピッカー コントラクト



このスクリーン ショットは、アプリの保存場所を表示するページが読み込まれるファイル ピッカー ウィンドウの中央の領域を強調表示したものです。

- **Windows や他のアプリからユーザーがアクセスできない保存場所を提供する**
  Windows や他のアプリから簡単にアクセスできない場所 (アプリの保存フォルダー、リモートの保存場所など) にユーザーがファイルを保存できるようにします。
- **選ばれたファイルの種類に基づいて、ファイル ピッカーのページに表示されるファイルを変更する**
  ユーザーがファイル ピッカーのファイルの種類のドロップダウン リストでファイルの種類を変更した場合、選ばれたファイルの種類に一致するファイルだけを表示するようにビューを更新する必要があります。表示されるファイルを種類で絞り込むと、ユーザーが目的の種類のファイルを一貫した方法で簡単に見つけられるようになります。
- **アプリのファイル ピッカーのページでファイルを選択することで、ユーザーがそのファイルを簡単に置き換えることができるようにする**
  ユーザーがファイル ピッカーのページでファイルを選択する場合は、既存のファイルを簡単に置き換えることができるように、ファイル ピッカーのファイル名ボックスでそのファイルの名前を自動的に置き換える必要があります。

追加の UX ガイドライン: キャッシュ ファイル アップデーター コントラクト

- **ファイルの追跡と更新に対応したリポジトリを提供する**
  ユーザーがアプリを主な保存場所 (ユーザーがファイルを保存したりファイルにアクセスしたりするために通常使う場所) として使う場合は、アプリで一部のファイルを追跡してリアルタイム更新を提供することができます。
- **堅牢なリポジトリを提供するようにアプリとファイル ピッカーのページを設計する**
  ユーザーがアプリをファイルの主な保存場所として使う場合は、アプリと、関連するファイル ピッカーのビューを、ファイルの頻繁な更新やバージョンの競合などによるデータ消失を防ぐように設計します。
- **更新中に発生した問題をユーザーが解決できるようにする**
  更新を正常に完了できるようにするには、ファイルの更新中または保存中に発生した問題を効果的に解決するためにユーザーが操作する必要がある場合に、(**UIRequested** を使って) リアルタイムにユーザーに通知する必要があります。資格情報、ファイルのバージョンの競合、ディスク容量の問題の解決をサポートすることは特に重要です。作成する UI は、軽量で、問題の解決に特化したものにする必要があります。 複数の手順が必要な場合は (ログインなど)、すべての手順をアプリのファイル ピッカーのページで処理する必要があります。 完了したら、アプリでファイル ピッカーのコミット UI を有効にすることができます。アプリでは、ファイル ピッカーのタイトルを更新して、ユーザーに現在の場所についての情報を示す必要もあります。
  ユーザーが問題をリアルタイムで解決できない場合や、ユーザーに状況を通知するだけでよい場合 (ユーザーには解決できないエラーが発生した場合など) は、問題の発生時に **UIRequested** を使ってすぐに通知するのではなく、アプリの次回起動時に通知することをお勧めします。
- **更新操作と保存操作に関する追加情報をアプリの通常のページで提供する**
  メイン アプリの UI で、ユーザーが進行中の操作や今後の操作の設定を管理したり、進行中の操作や以前の操作に関する情報を確認したり、発生したエラーに関する情報を入手したりできるようにする必要があります。

ファイル ピッカーのガイドライン

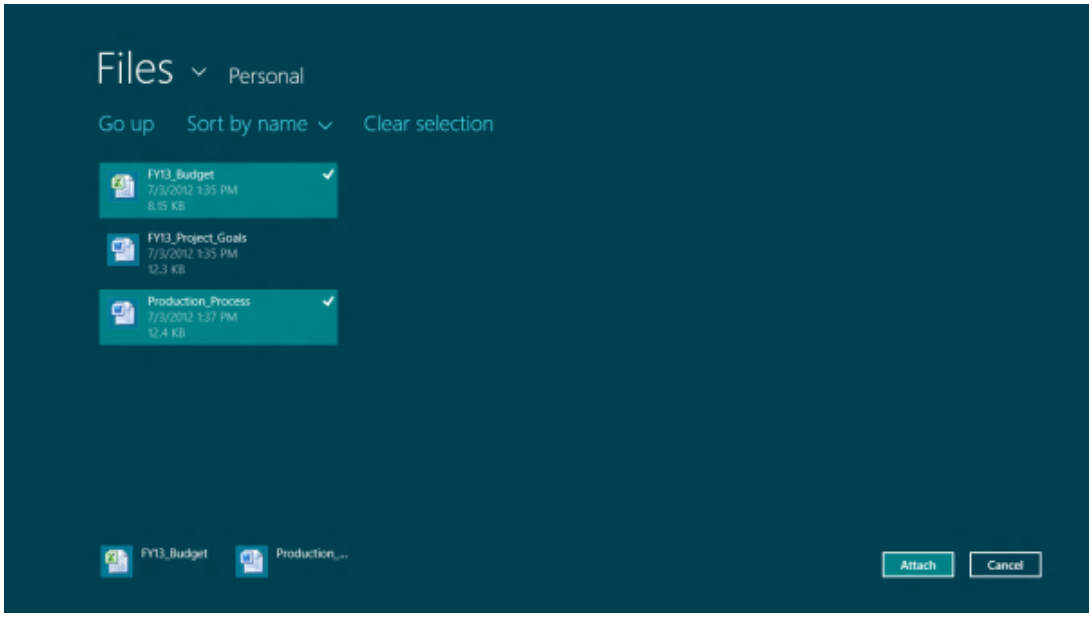
アプリは、ファイル ピッカーを使って、ファイルとフォルダーにアクセスし、ファイルを保存できます。

推奨と非推奨

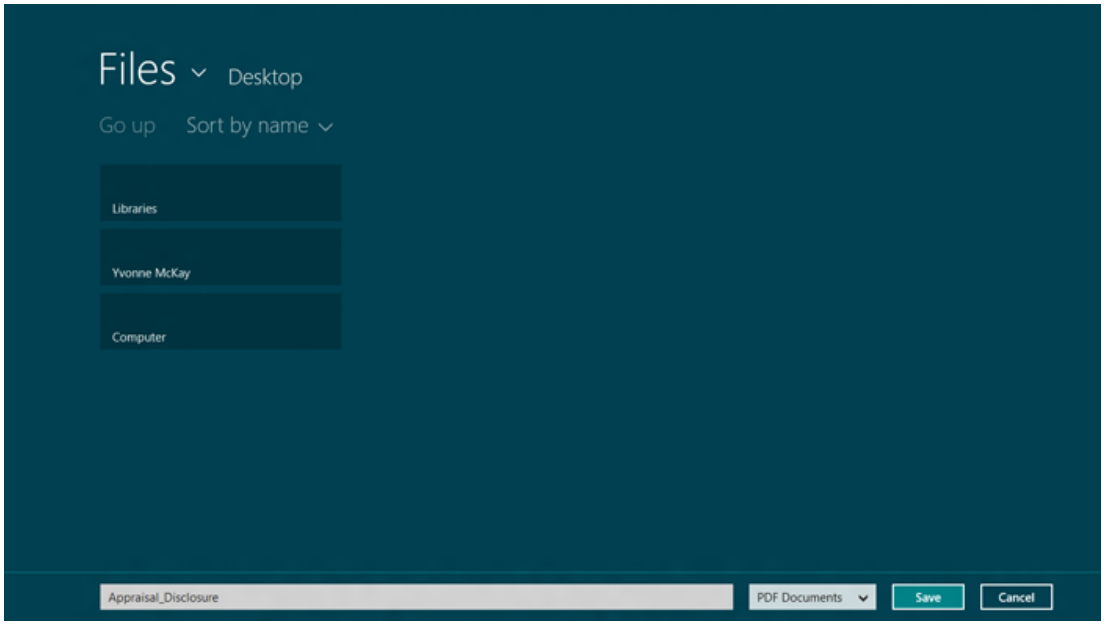
- ファイル ピッカーを呼び出すコントロールをアプリに追加して、ユーザーがアプリで操作するファイルを選ぶことができるようにします。
- ファイル ピッカーを呼び出すコントロールをアプリの UI に追加して、(他のアプリと同様に) ユーザーが保存するファイルの名前、種類、保存場所を指定できるようにします。
- アプリで処理できるファイルの種類だけをユーザーが選択または保存できるようにファイルの種類を設定する。
- ファイルまたはフォルダーにアクセスする場合は、ユーザーが選ぶ項目の種類に基づいて表示モードを設定する
- コミット ボタンにユーザーの現在の作業に対応するテキストを設定する
- 提示する開始場所は、ユーザーの現在の作業に基づいてできるだけ関連性のある場所に設定する
- ファイルにアクセスする場合は、ユーザーが現在の作業に応じて 1 つのファイルまたは複数のファイルを選べるようにする
- ファイルを保存する場合は、保存するファイルの既定のファイル名を設定する
- ファイルの内容の参照、使用、管理にファイル ピッカーを使わない
- ユーザーが独自の名前や場所を指定する必要がない場合は、ファイルの保存にファイル ピッカーを使わない

その他の使い方のガイダンス

- **ファイルとフォルダーにアクセスする。**
  ファイル ピッカーを呼び出すコントロールをアプリに追加して、ユーザーがアプリで操作するファイルを選ぶことができるようにします。ユーザーは、次のスクリーン ショットに示すようなファイル ピッカーの UI を使って、ファイルの選択ができます。
  たとえば、次のスクリーン ショットは、ユーザーがファイルを選べるように呼び出されたファイル ピッカーを示しています。スクリーン ショットでは、ユーザーが 2 つのファイルを選んでいます。



  ユーザーは、ファイル ピッカーの左上にあるドロップダウン リストに示される任意の場所 (他のアプリを含む) からファイルを選ぶことができます。
- **アプリに "名前を付けて保存" を追加する**
  ファイル ピッカーを呼び出すコントロールをアプリの UI に追加して、(他のアプリと同様に) ユーザーが保存するファイルの名前、種類、保存場所を指定できるようにします。ユーザーは、次のスクリーン ショットに示すようなファイル ピッカーの UI を使って、ファイルの移動や保存ができるようになります。



アプリに専用のページと UI を作って、ユーザーがファイルの内容を参照、使用、管理できるようにすることをお勧めします。これにより、ユーザーは現在の作業に集中できるようになり、ファイルを選ぶときに不要な機能に惑わされることがなくなります。

たとえば、フォト ギャラリー アプリでは、アプリで画像ファイルを整理したり表示したりするためのカスタマイズされた専用のページと UI を用意します。この UI は、ユーザーのニーズに合わせてアプリでカスタマイズできます。ユーザーがファイルをギャラリーに追加するときは、選択操作に特化したエクスペリエンスを提供するファイル ピッカーをアプリで呼び出します。

ユーザーがファイル名、ファイルの種類、または保存場所を指定する必要がない場合は、(ファイル ピッカーを起動せずに) アプリで自動的にファイルをバックグラウンド保存することをお勧めします。このようにすると不必要なユーザー操作がなくなるため、ファイルの保存プロセスが迅速になり、煩わしさも減ります。

ユーザー エクスペリエンス ガイドライン: ファイルとフォルダーへのアクセスと保存

- ファイルまたはフォルダーを選ぶ場合も保存する場合も、アプリがサポートし、ユーザーの現在の作業に関連するファイルの種類だけを表示するように、ファイル ピッカーをカスタマイズします。たとえば、ユーザーが動画を選択または保存する場合、アプリで処理できる形式を使った動画ファイルだけを選択または保存できるようにファイルの種類を設定します。

これは、ユーザーがファイル ピッカーで表示されるファイルを使っているときにフォルダーを選ぶ際にも適用されるため、選ぶフォルダーを決めるのに役立ちます。 適切なファイルの種類を表示するようフィルター処理を行うことで、ユーザーは正しいフォルダーをより速く特定できます。

- ユーザーが画像または動画を選ぶ場合は、表示モードを **Thumbnail** に設定します。 ユーザーがその他の種類のファイルまたはフォルダーを選ぶ場合は、表示モードを **List** に設定します。

  場合によっては、ユーザーが画像 / 動画やその他の種類のファイルを選ぶこともあります (たとえば、ユーザーが電子メールに添付したり IM で送信したりするファイルを選ぶ場合)。 この場合は、アプリに 2 つの UI コントロールを追加して、両方の表示モードをサポートする必要があります。1 つのコントロールは **Thumbnail** 表示モードを使ってファイル ピッカーを呼び出して、ユーザーが画像や動画を選べるようにし、もう 1 つのコントロールは **List** 表示モードを使ってファイル ピッカーを呼び出して、ユーザーが他の種類のファイルを選べるようにします。たとえば、メール アプリには、**[画像または動画の添付]** と **[ドキュメントの添付]** という 2 つのボタンを用意します。
- ファイルまたはフォルダーを選ぶ場合も保存する場合も、コミット ボタンにユーザーの現在の作業に適したテキストを設定してファイル ピッカーをカスタマイズします。たとえば、ユーザーがアプリにアップロードする一連のファイルを選ぶ場合は、コミット ボタンのテキストを "アップロード" に設定します。
- ファイルまたはフォルダーを選ぶ場合も保存する場合も、ユーザーの現在の作業と **PickerLocationId** 列挙体で提供される有効な開始場所のリストに基づいて、できるだけ関連性のある開始場所を提示するようにファイル ピッカーをカスタマイズします。たとえば、ユーザーが画像を選ぶ場合は、提示する開始場所をユーザーのピクチャに設定します。
- ユーザーがプロフィール画像を選ぶ場合は、1 つのファイルを選ぶファイル ピッカーを呼び出します。ユーザーが友人に送る写真を選ぶ場合は、複数のファイルを選ぶファイル ピッカーを呼び出します。
- ユーザーが用意されている既定のファイル名をそのまま使うと、別の名前を入力する手間が省けるため、"名前を付けて保存" 作業を速やかに完了できます。既定のファイル名を設定するには、**FileSavePicker.SuggestedFileName** プロパティを使います。

ジオフェンスのガイドライン

Windows ランタイム アプリの **geofencing** については、次のベスト プラクティスに従ってください。

推奨と非推奨

- **Geofence** イベントが発生したときにインターネット アクセスが必要な場合は、ジオフェンスを作成する前にインターネット アクセスを確認します。
  - アプリで現在インターネットにアクセスできない場合、ジオフェンスをセットアップする前にユーザーに対してインターネットに接続するようメッセージを表示することができます。
  - インターネット アクセスが不可能である場合は、ジオフェンスの位置確認に必要な電力を消費しないようにしてください。
- ジオフェンス イベントが **Entered** 状態または **Exited** 状態に対する変更を示す場合、タイム スタンプと現在の位置をチェックしてジオフェンス通知の関連性を確認します。詳しくは、次の「タイム スタンプと現在位置の確認」をご覧ください。
- デバイスで位置情報にアクセスできない場合は、ケースを管理する例外を作成し、必要に応じてユーザーに通知します。アクセス許可がオフになっている、デバイスに GPS 機能が付いていない、GPS 信号がブロックされている、Wi-Fi 信号が弱いなどの理由で、位置情報が利用できない場合があります。
- 一般に、フォアグラウンドとバックグラウンドの両方で同時にジオフェンス イベントをリッスンする必要はありません。ただし、アプリがフォアグラウンドとバックグラウンドの両方で同時にジオフェンス イベントをリッスンする必要がある場合は、次の手順を行います。
  - **ReadReports** メソッドを呼び出して、イベントが発生したかどうかを確認します。
  - ユーザーからアプリが見えなくなったときはフォアグラウンド イベント リスナーの登録を解除し、再び見えるようになったときにもう一度登録します。

  コード例と詳しい情報については、「バックグラウンドとフォアグラウンドのリスナー」をご覧ください。

- 1 つのアプリに 1000 以上のジオフェンスを使わないでください。システムは実際にはアプリごとに数千のジオフェンスをサポートしますが、1000 以下のジオフェンスを使用することによってアプリのメモリ使用量を減らしてアプリの高パフォーマンスを維持できます。
- 半径が 50 m 未満のジオフェンスを作成しないでください。 アプリで小さなジオフェンスを使う必要がある場合は、最高のパフォーマンスを実現するために GPS 機能付きのデバイスでアプリを使うようユーザーに勧めてください。

その他の使い方のガイダンス

タイム スタンプと現在位置の確認

イベントにより **Entered** または **Exited** 状態に変化したときは、イベントのタイム スタンプと現在位置の両方を確認します。 イベントが実際にユーザーによって処理される時期は、システムでバックグラウンド タスクを起動するリソースが不足していたり、ユーザーが通知に気付かなかったり、デバイスがスタンバイ中であったり (Windows の場合) など、さまざまな要因によって影響を受けます。たとえば、次のような順序で事態が進む可能性があります。

- アプリがジオフェンスを作成して、ジオフェンスの進入イベントと退出イベントを監視します。
- ユーザーがデバイスをジオフェンスの内側に移動して、進入イベントをトリガーします。
- ジオフェンスの内側に入ったという通知をアプリがユーザーに送信します。
- ユーザーが忙しく、通知に気付いたのは 10 分後でした。
- その 10 分間の間に、ユーザーはジオフェンスの外側に移動していました。

タイム スタンプをみれば、アクションが過去に起こったことを判断できます。 現在位置をみれば、ユーザーがジオフェンスの外側に戻ったことが確認できます。アプリの機能によっては、このイベントを無視することもできます。

バックグラウンドとフォアグラウンドのリスナー

一般に、アプリは、フォアグラウンド タスクとバックグラウンド タスクの両方で同時に **Geofence** イベントをリッスンする必要はありません。 両方が必要になる場合に最も明確な処理方法は、バックグラウンド タスクに通知処理を任せることです。 実際にフォアグラウンドとバックグラウンドの両方でジオフェンス リスナーをセットアップした場合、どちらが最初にトリガーされるか不明であるため、常に **ReadReports** メソッドを呼び出してイベントが発生したか確認する必要があります。

また、フォアグラウンドとバックグラウンドの両方でジオフェンス リスナーをセットアップした場合、ユーザーからアプリが見えなくなるたびにフォアグラウンド イベント リスナーの登録を解除し、再び見えるようになったときにアプリを再登録する必要があります。表示イベントに登録するコード例を次に示します。

```
document.addEventListener("visibilitychange", onVisibilityChanged, false);
```

```
    Windows.UI.Core.CoreWindow coreWindow;

    coreWindow = CoreWindow.GetForCurrentThread(); // This needs to be set before
InitializeComponent sets up event registration for app visibility
    coreWindow.VisibilityChanged += OnVisibilityChanged;
```

表示が変わると、ここで示したようにフォアグラウンド イベント ハンドラーの有効または無効を切り替えることができます。

```
function onVisibilityChanged() {
    // NOTE: After the app is no longer visible on the screen and before the app is
suspended
    // you might want your app to use toast notification for any geofence activity.
    // By registering for VisibiltyChanged the app is notified when the app is no longer
visible in the foreground.

    if (document.msVisibilityState === "visible") {
        // register for foreground events

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.addEventListener("geo
fencestatechanged", onGeofenceStateChanged);

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.addEventListener("stat
uschanged", onGeofenceStatusChanged);
    } else {
        // unregister foreground events (let background capture events)
```



```
Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.removeEventListener("
geofencestatechanged", onGeofenceStateChanged);

Windows.Devices.Geolocation.Geofencing.GeofenceMonitor.current.removeEventListener("
statuschanged", onGeofenceStatusChanged);
    }
}
```

```
private void OnVisibilityChanged(CoreWindow sender, VisibilityChangedEventArgs args)
{
    // NOTE: After the app is no longer visible on the screen and before the app is
suspended
    // you might want your app to use toast notification for any geofence activity.
    // By registering for VisibiltyChanged the app is notified when the app is no longer
visible in the foreground.

    if (args.Visible)
    {
        // register for foreground events
        GeofenceMonitor.Current.GeofenceStateChanged += OnGeofenceStateChanged;
        GeofenceMonitor.Current.StatusChanged += OnGeofenceStatusChanged;
    }
    else
    {
        // unregister foreground events (let background capture events)
        GeofenceMonitor.Current.GeofenceStateChanged -= OnGeofenceStateChanged;
        GeofenceMonitor.Current.StatusChanged -= OnGeofenceStatusChanged;
    }
}
```

ジオフェンスのサイズ変更

GPS を使うと最も正確な位置情報が得られますが、ジオフェンスでは Wi-Fi などの位置センサーを使ってユーザーの現在位置を判断することもできます。 しかし、GPS とは別のこういった方法を使うと、作成できるジオフェンスのサイズが影響を受けます。 精度が低い場合、小さなジオフェンスを作成しても役に立ちません。 通常、50 m より半径が小さいジオフェンスを作らないことをお勧めします。また、Windows ではジオフェンスのバックグラウンド タスクが周期的にしか実行されないため、小さなジオフェンスを使った場合、**Enter** イベントや **Exit** イベントをまったく認識できない可能性があります。

アプリで小さなジオフェンスを使う必要がある場合は、最高のパフォーマンスを実現するために GPS 機能付きのデバイスでアプリを使うようユーザーに勧めてください。

位置認識アプリのガイドライン

このトピックでは、位置情報認識アプリを構築する際のパフォーマンス ガイドラインを説明します。

推奨と非推奨

- location オブジェクトは、アプリで位置データが必要になった場合にのみ使用を開始します。
  アプリが **Geolocator** オブジェクトに初めてアクセスする際には、同意のプロンプトが表示されます。これは、アプリが **getGeopositionAsync** を初めて呼び出すか、 **positionChanged** イベントのイベント ハンドラーを初めて登録することで発生します。アプリの主要目的にとって位置データへのアクセスが必要ない場合には、アプリが起動してすぐにデバイスを使うためのアクセス許可を確認するプロンプトが表示されると、ユーザーにとって紛らわしくなる可能性があります。
- アプリで位置情報が必須でない場合は、位置情報を必要とするタスクをユーザーが完了することを試みるまではその情報にアクセスしないでください。たとえば、ソーシャル ネットワーキング アプリに、[位置情報を使ってチェックイン] というボタンがある場合、アプリは、ユーザーがそのボタンをクリックするまでは位置情報にアクセスしないようにします。アプリのメイン機能で位置情報が必要な場合は、すぐにアクセスしても問題ありません。
- Windows 8 でのみ、ユーザーに同意のプロンプトを表示するために、**Geolocator** オブジェクトの初めての使用はメイン UI スレッドで行う必要があります。
  Geolocator の初めての使用は、**getGeopositionAsync** の初めての呼び出し、または **positionChanged** イベントのハンドラーの初めての登録にすることができます。 同意のプロンプトについては、「機密性の高いデバイスの使用のガイドライン」で詳しく説明されています。つまり、JavaScript を使うアプリでは、

Geolocator オブジェクトを初めて使う際には、アクティブ化ハンドラーでこのオブジェクトを使うことはできません。

- 位置データがどのように使われるかをユーザーに知らせてください。
- ユーザーが現在の位置を手動で更新できる UI を用意します。
- 位置データの取得中は、進行状況バーまたは進行状況リングを表示します。使えるプログレス コントロールとその使い方について詳しくは、「プログレス コントロールのガイドライン」をご覧ください。
- 位置情報サービスが無効または利用不可になっている場合は、適切なエラー メッセージまたはダイアログを表示します。

  ユーザーが位置データへのアクセスを無効にしている場合、または他の理由によりアプリでデータを利用できない場合は、ユーザーに対して適切なエラー メッセージを表示します。

  - Windows でのお勧めのメッセージは、"位置情報サービスは現在無効になっています。設定チャームを使って位置情報サービスを有効にしてください" です。Windows Phone では、次のメッセージを使用できます。"位置情報はデバイスでは無効になります。位置情報を有効にするには、設定に移動して位置を選択してください。"
  - エラー メッセージがアプリのフローを妨げないようにします。位置データがアプリにとって必須ではない場合は、インライン テキストとしてメッセージを表示します。ソーシャル ネットワーキング アプリやゲーム アプリがこのカテゴリに分類されます。
  - 位置データがアプリの機能にとって必須の場合は、ポップアップやダイアログとしてメッセージを表示します。地図アプリやナビゲーション アプリがこのカテゴリに分類されます。
  - プログラムを使って設定チャームを表示しようとしないでください。

  インライン、ダイアログ、ポップアップのエラー メッセージの例については、「個人データにアクセスするデバイスのガイドライン」および「UI のレイアウト」の「**エラー**」をご覧ください。

- 位置情報へのアクセスをユーザーが無効にした場合は、キャッシュされた位置データをクリアし、**Geolocator** オブジェクトを解放します。

ユーザーが設定を使って位置情報へのアクセスをオフにした場合に、**Geolocator** オブジェクトを解放します。すると、アプリは、あらゆる位置情報 API 呼び出しの結果として **ACCESS_DENIED** を受け取ります。アプリで位置データを保存またはキャッシュしている場合は、ユーザーが位置情報へのアクセスを無効にするときにすべてのキャッシュ データをクリアします。位置情報サービス経由で位置データを利用できないときに位置情報を手動で入力するための代替手段を用意してください。

- 位置情報サービスを再び有効にするための UI を用意します。たとえば、**Geolocator** オブジェクトを再インスタンス化して位置情報を取得し直す更新ボタンを提供します。

  位置情報サービスを再び有効にするための UI を提供するー

  - ユーザーが位置情報を無効にした後に再び有効にした場合、アプリには通知されません。**status** プロパティは変更されず、**statusChanged** イベントも発生しません。アプリで、新しい **Geolocator** オブジェクトを作成し、**getGeopositionAsync** を呼び出して更新された位置情報データを取得するか、**positionChanged** イベントの受信登録をもう一度行います。位置情報が再び有効になったことを確認できたら、位置情報サービスが無効であることをユーザーに通知するために表示していた UI をクリアし、新しい状態に対して適切に対応します。
  - アプリをアクティブ化するとき、位置情報が必要な機能をユーザーが明示的に使おうとしたときなど、状況に応じて必要と思われる任意の時点で、位置情報データを取得し直すことをお勧めします。

パフォーマンス

- アプリで位置情報の更新を受け取る必要がない場合は、位置情報の要求を 1 回だけ使います。たとえば、写真に位置情報タグを追加するアプリでは、位置情報更新イベントを受け取る必要はありません。このようなアプリでは、**getGeopositionAsync** を使って位置情報を要求します。詳しくは、「ユーザーの位置の検出」をご覧ください。

  1 回限りの位置情報の要求を行う場合は、次の値を設定する必要があります。

- **DesiredAccuracy** または **DesiredAccuracyInMeters** を設定して、アプリから要求される精度を指定します。これらのパラメーターを使用する場合の推奨事項については、以下をご覧ください
- **getGeopositionAsync** の最大保存期間のパラメーターを設定して、アプリで有用な位置情報を取得できる期間を指定します。アプリで数秒または数分前の位置を使用できる場合は、ほとんどすぐに位置を受け取って、デバイスの電力を節約することができます。
- **getGeopositionAsync** のタイムアウト パラメーターを設定します。これが、アプリが返される位置またはエラーを待機することができる長さです。ユーザーへの応答性とアプリが必要とする精度のバランスを理解する必要があります。

- 頻繁に位置を更新する必要がある場合は、連続的な位置情報のセッションを使います。特定のしきい値を超えた移動を検出する場合、または発生時に絶えず位置情報の更新を取得する場合は、**positionChanged** イベントと **statusChanged** イベントを使います。

  位置情報の更新を要求すると、**DesiredAccuracy** または **DesiredAccuracyInMeters** を設定して、アプリから要求される精度を指定する必要があります。また、**MovementThreshold** または **ReportInterval** を使って、位置情報の更新が必要な頻度を設定する必要があります。

  - 移動しきい値を指定します。アプリによっては、ユーザーの移動距離が大きいときにだけ位置情報を更新すれば済むものがあります。たとえば、地域のニュースや天気予報の更新情報を提供するアプリでは、ユーザーの位置が別の都市に変わらない限り位置情報を更新する必要はありません。このような場合は、**MovementThreshold** プロパティを設定して、位置情報更新イベントの発生条件となる最小の移動距離を調整します。このプロパティには **PositionChanged** イベントをフィルター処理する効果があります。これらのイベントは、位置の変化が移動しきい値を超えたときにのみ発生します。Windows では、**MovementThreshold** プロパティを設定しても、位置データの提供元 (Windows 位置情報取得機能や接続された GPS デバイスなど) で位置情報が計算される頻度は変わりません。Windows Phone では、移動の予想速度などのその他の要因と共に **MovementThreshold** プロパティを使って、デ

バイスの電力を節約するために、システムで計算される位置情報の頻度を調整します。

- アプリのエクスペリエンスと整合し、システム リソースの使用が最小限に抑えられる **ReportInterval** を使います。 たとえば、天気予報アプリでは、15 分ごとにデータを更新するだけでよいと思われます。リアルタイムのナビゲーション アプリを除くほとんどのアプリでは、位置情報の更新について、高い精度のストリームを常に必要とするわけではありません。最大限の精度のデータストリームを必要としない場合や、頻繁に更新する必要がない場合は、**ReportInterval** プロパティを設定して、アプリで位置情報を更新する必要がある最小の頻度を指定します。これにより、必要なときにだけ位置情報を計算することで、位置情報の提供元の電力を節約できます。

  リアルタイムのデータを必要とするアプリでは、最短の間隔を指定せずに、**ReportInterval** を 0 に設定する必要があります。Windows では、レポート間隔が 0 の場合、アプリは最も精度が高い位置情報の提供元から送られる頻度でイベントを受け取ります。Windows Phone では、アプリから要求された精度に依存するで速度で更新を取得します。

  位置データを提供するデバイスでは、さまざまなアプリから要求されるレポート間隔を追跡し、要求された最短の間隔でデータをレポートする場合があります。これにより、精度の要件が最も高いアプリに必要なデータを提供できます。そのため、別のアプリで要求された更新頻度の方が高い場合は、要求した頻度よりも頻繁に更新が生成されることがあります。

  注 位置情報の提供元からのレポート間隔は、必ずしも要求どおりになるとは限りません。 位置情報取得機能デバイスによってはレポート間隔を追跡しないものもありますが、追跡されるものとして指定しておくことをお勧めします。

- 電力を節約するには、**DesiredAccuracy** プロパティを設定して、アプリで高い精度のデータが必要かどうかを位置情報プラットフォームに示します。高い精度のデータを必要とするアプリがなければ、GPS 位置情報取得機能を無効にして電力を節約できます。
  - GPS でデータを取得するには、**DesiredAccuracy** を **HIGH** に設定します。
  - ターゲティング広告のためにのみ位置情報を使うアプリは、消費電力を最小限に抑えるため、**DesiredAccuracy** を **Default** に設定します。

精度についてアプリに特定のニーズがある場合は、**DesiredAccuracy** を使う代わりに **DesiredAccuracyInMeters** プロパティを使うこともあります。これは、通常、位置情報を移動体通信ビーコン、Wi-Fi ビーコンや衛星に基づいて取得できる Windows Phone に特に役立ちます。より具体的な精度値を選ぶと、システムが位置情報を提供する際に最も低い消費電力で適切なテクノロジを識別するために役立ちます。

たとえば、次のような場合があります

- アプリが広告の調整、天気、ニュースなどのための位置情報を取得している場合は、一般に 5000 m の精度で十分です。
- アプリが地域内のごく近隣を表示する場合は、結果の表示には一般に 300 m の精度が適しています。
- ユーザーがお勧めの近くのレストランを探している場合は、ブロック内の位置を取得する必要がありますので、100 m の精度で十分です。
- ユーザーが自身の位置を共有しようとしている場合は、アプリには約 10 m の精度が必要です。

◦ アプリに特定の精度の要件がある場合は **Geocoordinate.accuracy** プロパティを使います。たとえば、ナビゲーション アプリでは、**Geocoordinate.accuracy** プロパティを使って、利用可能な位置情報データがアプリの要件を満たしているかどうかを調べます。

- 起動時の待ち時間を考慮します。アプリで初めて位置データを要求したとき、位置情報取得機能が起動するまでに 1 ～ 2 秒の待ち時間が発生することがあります。アプリの UI を設計するときは、この点に注意してください。たとえば、**GetGeopositionAsync** の呼び出しを保留している他のタスクがブロックされないようにしてください。
- バックグラウンドの動作を考慮します。Windows ランタイム アプリにフォーカスがない場合、バックグラウンドで中断されている間は位置情報更新イベントを受け取りません。位置情報の更新をログに記録して追跡する場合は、この点に注意してください。アプリにフォーカスが戻った後は、新しいイベントだけを受け取ります。アプリが非アクティブだったときに発生した更新は取得されません。
- ロー センサーとフュージョン センサーを効率的に使います。 Windows 8 では、ローとフュージョンの 2 種類のセンサーがサポートされます。

- ロー センサーには、加速度計、ジャイロメーター、磁力計が含まれます。
- フュージョン センサーには、向き、傾斜計、コンパスが含まれます。フュージョン センサーは、ロー センサーの組み合わせからデータを取得します。

Windows ランタイム API は磁力計以外のすべてのセンサーにアクセスできます。フュージョン センサーの方がロー センサーよりも正確で安定していますが、より多くの電力を使います。用途に適したセンサーを使う必要があります。詳しくは、「使用に適したセンサーの選択」をご覧ください。

コネクト スタンバイ: Windows のみ。PC がコネクト スタンバイ状態にある場合、**Geolocator** オブジェクトはいつでもインスタンス化できます。しかし、**Geolocator** オブジェクトは集約する対象のセンサーを見つけることができず、**GetGeopositionAsync** の呼び出しは 7 秒後にタイムアウトします。**PositionChanged** イベント リスナーの呼び出しは行われず、**StatusChanged** イベント リスナーは 1 回呼び出され、そのステータスは **NoData** となります。

その他の使い方のガイダンス

位置情報設定の変更を検出する

Windows では、位置情報機能は、設定チャームやコントロール パネルを使ってユーザーが無効にできるようになっています。位置情報の設定を変更する UI について詳しくは、「位置情報の設定」をご覧ください。Windows Phone では、ユーザーは設定アプリで位置情報を使用できないようにすることができます。ユーザーによる設定変更の処理についての設計ガイダンスは、「機密性の高いデバイスの使用のガイドライン」をご覧ください。

- ユーザーが位置情報サービスを無効にしたり再び有効にしたことを検出するには、次の操作を行います。
  - **StatusChanged** イベントを処理します。**StatusChanged** イベントの引数である **Status** プロパティの値は、ユーザーが位置情報サービスを無効にすると **Disabled** になります。
  - **GetGeopositionAsync** から返るエラー コードをチェックします。ユーザーによって位置情報サービスが無効にされている場合、**GetGeopositionAsync** の呼び出しは **ACCESS_DENIED** エラーで失敗し、**LocationStatus** プロパティの値は **Disabled** になっています。

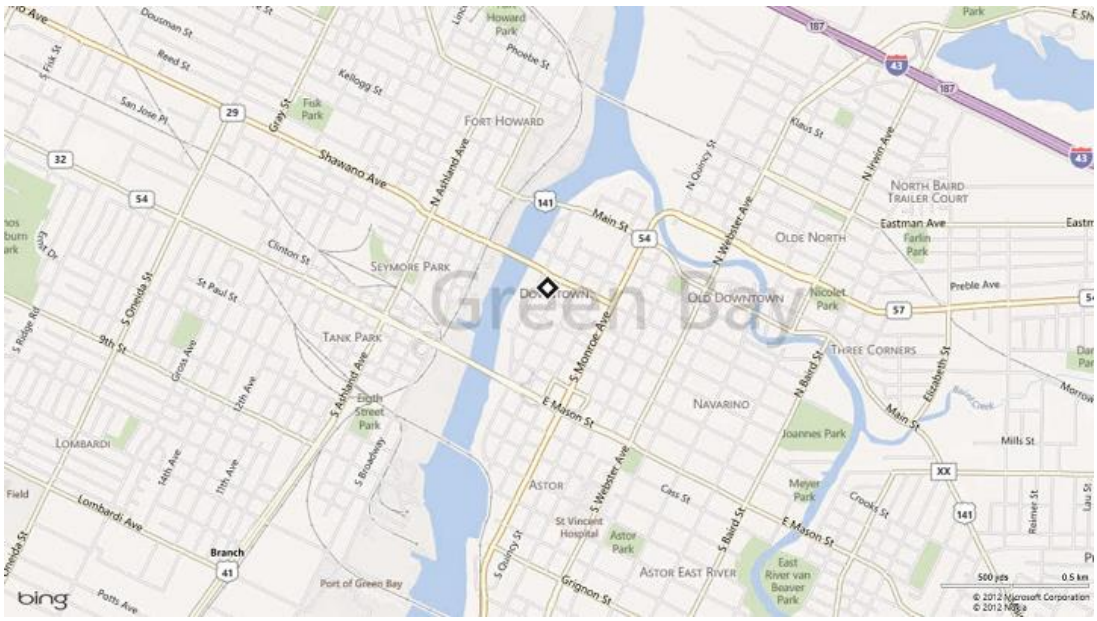
- 地図アプリのような、位置情報データが必須のアプリの場合は、必ず次の操作を実行してください。
  - ユーザーの位置情報が変わったときに更新情報を取得できるように、**PositionChanged** イベントを処理します。
  - 前の説明に従って **StatusChanged** イベントを処理し、位置設定の変化を検出します。詳しくは、「位置情報の更新に対応する方法」をご覧ください。

位置情報 API は、データが利用可能になったときにデータを返します。最初に誤差の範囲が大きい位置情報を返し、より正確な情報が利用可能になったときに位置情報を更新する場合があります。ユーザーの位置情報を表示するアプリでは、通常、より正確な情報が利用可能になったときに位置情報を更新する必要があります。

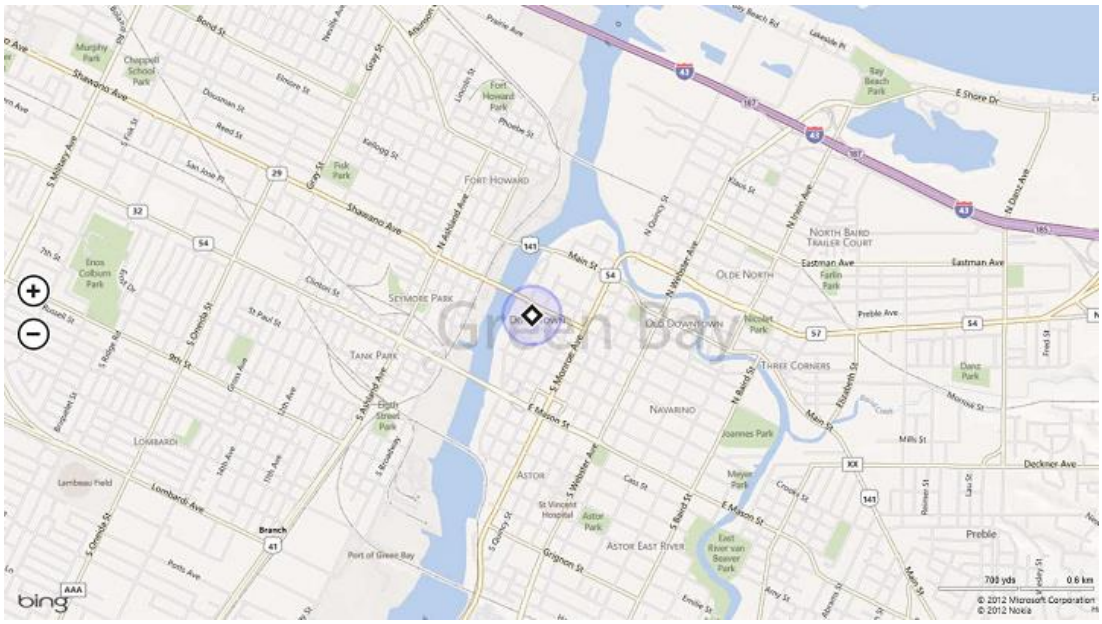
位置情報のグラフィックス表示

アプリでは、**Geocoordinate.accuracy** を使って、ユーザーの現在の位置情報を地図に明確に示すようにします。精度の幅は、主に、誤差の範囲が半径約 10 m、半径約 100 m、半径 1 km 超、という 3 種類があります。精度情報を使うことにより、アプリでは、利用可能なデータの状況に応じて位置情報を正確に表示することができるようになります。

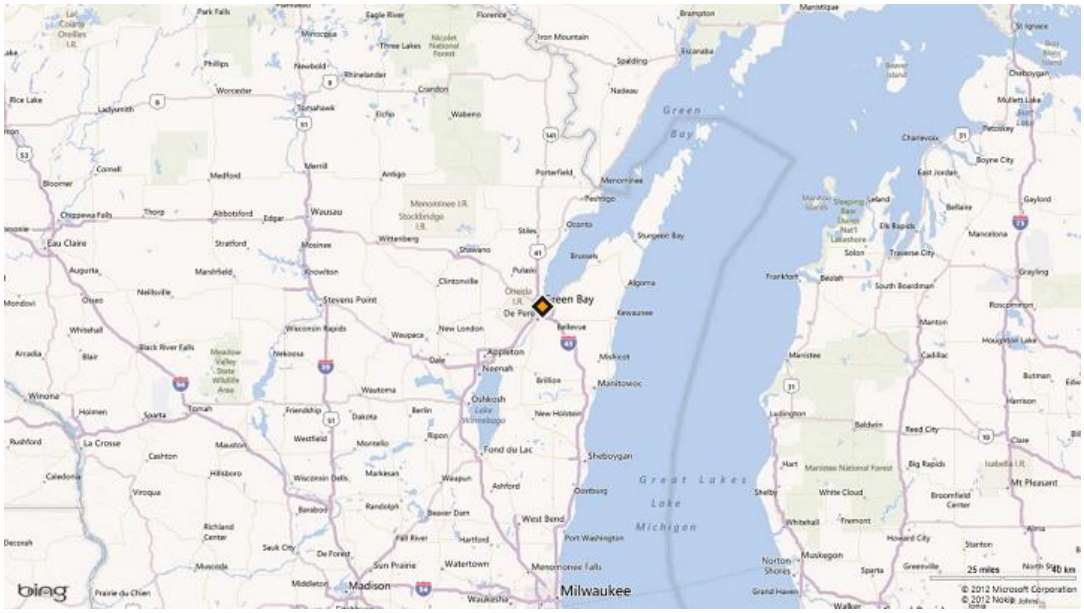
- 約 10 m 相当の精度 (GPS の解像度) の場合、位置情報は点またはピンで地図上に示すことができます。この精度では、経度と緯度の座標、住所の番地も表示できます。



- 10 ～ 500 m (おおよそ 100 m) の精度の場合、位置情報は通常、Wi-Fi による解決で受信されています。移動体通信から取得した位置情報は約 300 m の精度です。この場合、アプリでは誤差を含む範囲を表示することをお勧めします。道順を表示するアプリなど中心点が必要となる場合は、その中心点を誤差を表す範囲で囲むことができます。



- 戻される精度が 1 km より大きい場合は、IP レベルの解決で位置情報を受信することになります。多くの場合、地図上に特定の地点をピンポイントで表示するためにはこのレベルの精度は低すぎます。アプリではピンや円で地理的な場所を表現することは控え、その代わりに、地図を市のレベルまで、または誤差の範囲に応じて適切なエリア (たとえば、地域のレベル) までズームすることをお勧めします。



位置情報の精度が別の精度に切り替わるときは、異なるグラフィックス表示が適切に遷移するようにします。このためには、次のようにします。

- 切り替え時のアニメーションをスムーズにし、切り替えを高速かつ滑らかに保ちます。
- 数回の連続的な報告があるのを待ってから、精度が変わったと判断します。これにより、不要なズームが頻繁に行われるのを防ぐことができます。

位置情報のテキスト表示

天気アプリや地域情報アプリなどアプリの種類によっては、さまざまな精度の位置情報をテキストで表現することが必要になります。位置情報は、データが提供する精度レベルまでに抑えて、明確に表示するようにします。

- 精度が約 10 m 相当 (GPS の解像度) の場合、受信した位置情報データは相当に正確であるので、ごく近隣の地名のレベルで情報を伝えることができます。市の名前、都道府県の名前、国/地域の名前も使うことができます。
- 精度が約 100 m 相当 (Wi-Fi の解像度) の場合、受信した位置情報データはある程度正確です。市の名前までの情報を表示することをお勧めします。それより詳しい、近隣地域の地名の使用は避けてください。
- 1 km 超の精度 (IP による解決) の場合は、都道府県の名前、または国/地域の名前のみ表示します。

プライバシーに関する考慮事項

ユーザーの地理的な位置情報は、個人を特定できる情報 (PII) に当たります。ユーザーのプライバシーの保護に関するガイダンスについては、次の Web サイトをご覧ください。

- Microsoft のプライバシー

印刷のガイドライン

Windows ストア アプリでコンテンツを印刷する場合は、以下のガイドラインに従ってください。

推奨と非推奨

- 印刷ウィンドウでユーザーが無効な情報を入力したときには、エラー メッセージを表示します。修正操作を具体的に示し、エラー メッセージを 2 行以下に抑えます。
- 具体的なタスクに必要な場合を除き、アプリに印刷ボタンを使うことは避けます。一般に、ユーザーはデバイス チャームを使って印刷する必要があります。ただし状況によっては、印刷ボタンを追加することで、ユーザーのエクスペリエンスが簡素化される可能性があります。 たとえば、航空機へのチェックインの後に搭乗券を印刷するボタンが表示されるのがユーザーから見て自然な動作だとすると、印刷チャームを使うとタスクを複雑になることがあります。
- 印刷ウィンドウに表示される設定の順序を変更しないでください。ユーザーに表示される設定の順序はカスタマイズ可能ですが、設定を既定の順序を保持することで、エクスペリエンスの一貫性が維持されます。たとえば、既定の印刷エクスペリエンスでは [部数] の設定は最初に表示されているため、ユーザーはこの表示順序をアプリの印刷エクスペリエンスでも想定します。
- 絶対に必要でない限り、印刷ウィンドウにプリンター設定を追加しないでください。代わりに、プリンターの製造元がプリンター固有の設定を追加できるようにします。製造元によってプリンター固有の設定が提供されている場合、ユーザーは印刷ウィンドウの [その他の設定] をクリックして、追加の設定を表示できます (この表示を有効にする Windows ストア デバイス アプリがインストールされている場合)。

開発者向け

- **PrintTaskRequested** イベント ハンドラーでは、印刷タスクを作成するための最小限の処理を実行します。印刷可能なコンテンツを取得するために **PrintTaskSourceRequestedHandler** が呼び出されたときに備えて、より負荷の高い処理は残しておきます。

- 印刷をサポートしていないアプリのページに、**PrintTaskRequested** イベントを登録しないでください。**PrintTaskRequested** が登録されている場合、Windows は印刷がサポートされ、ユーザーが印刷できると想定します。たとえば、ニュース アプリがランディング ページからの印刷はサポートせず、コンテンツ ページからの印刷はサポートしている場合、ランディング ページの表示中は **PrintTaskRequested** を登録しないでください。

注 再利用できる URL を使うとき、Blob コンテンツは完全な忠実度でのみ印刷できます。詳細については、「ファイル システムへのアクセスの効率化」を参照してください。

印刷 UI 設計のガイドライン

このトピックでは、Windows ストア デバイス アプリに関連付けられた印刷 UI について説明します。 この種類のアプリは、デバイスに固有の補助的なエクスペリエンスをユーザーに提供します。特定のメーカーとモデルの印刷デバイスに固有の機能を強調すると、より豊かなユーザー エクスペリエンスを提供できます。このトピックに含まれる情報は、印刷デバイスと直接通信を行うアプリを作成する独立系ハードウェア ベンダーまたは開発者向けです。Windows ストア デバイス アプリのカスタマイズされた印刷 UI を設計するときは、次のガイドラインに従ってください。

デバイス アプリの作成方法については、ハードウェア デベロッパー センターの「プリンター用 Windows ストア デバイス アプリ」をご覧ください。

デバイス アプリでない印刷機能を備えたアプリを作成している場合は、「印刷のガイドライン」でより適切な推奨事項をご覧ください。

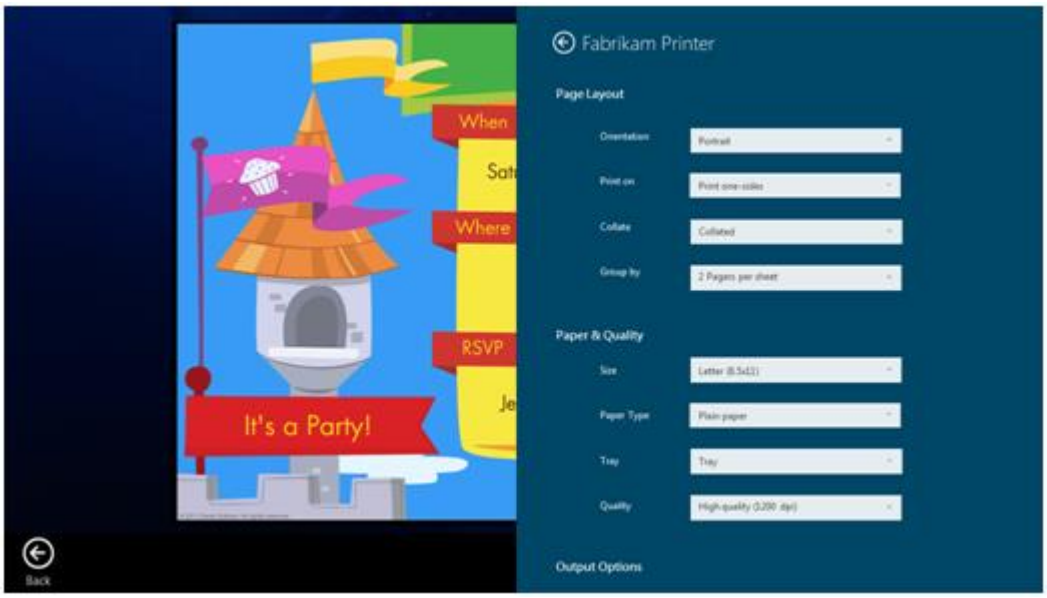
例

ここでは、Windows ストア デバイス アプリが、ユーザーの印刷エクスペリエンスを向上させる方法を例を挙げて示します。 このアプリでは、Windows が提供する既定の印刷エクスペリエンスの代わりに、カスタマイズされた **[More Settings]** ポップアップと **notification**

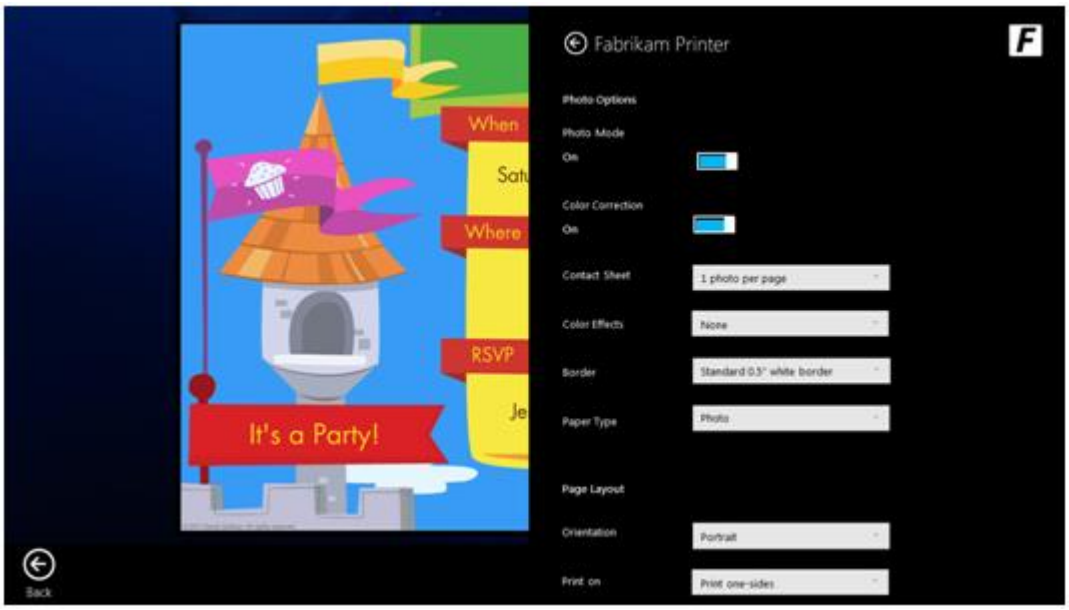
が用意され、それらをデバイスで使うことで、ユーザーやアプリに対しデバイス関連の警告を提供できます。

Amelia は、招待状を作成し、**[Print]** を選びます。

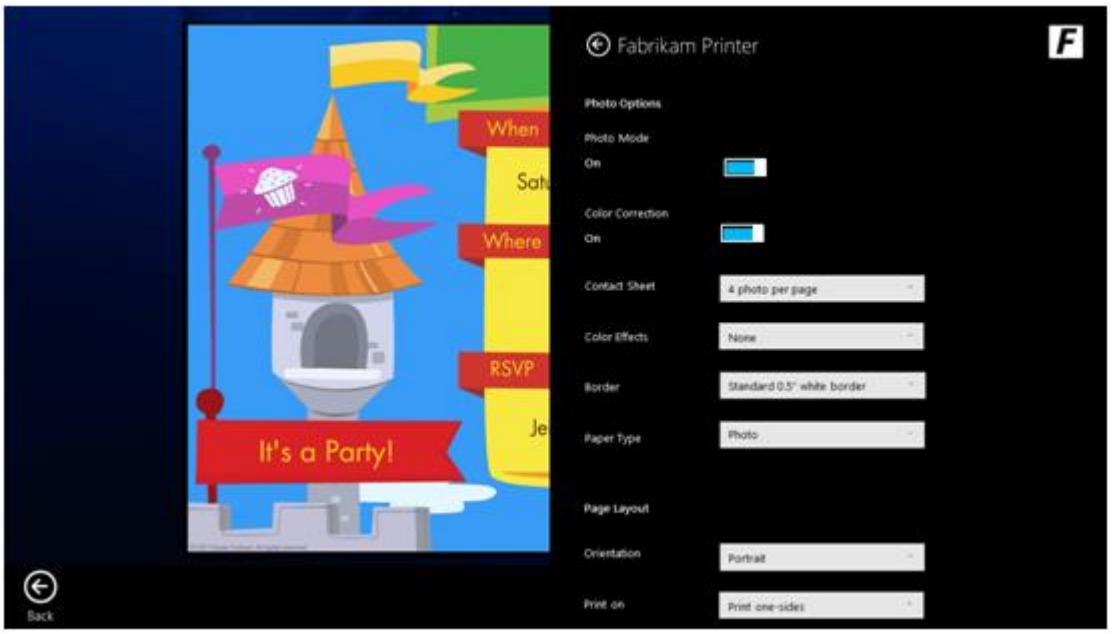
Amelia は、現在の印刷設定を確かめるために印刷ウィンドウで **[More Settings]** を選びます。



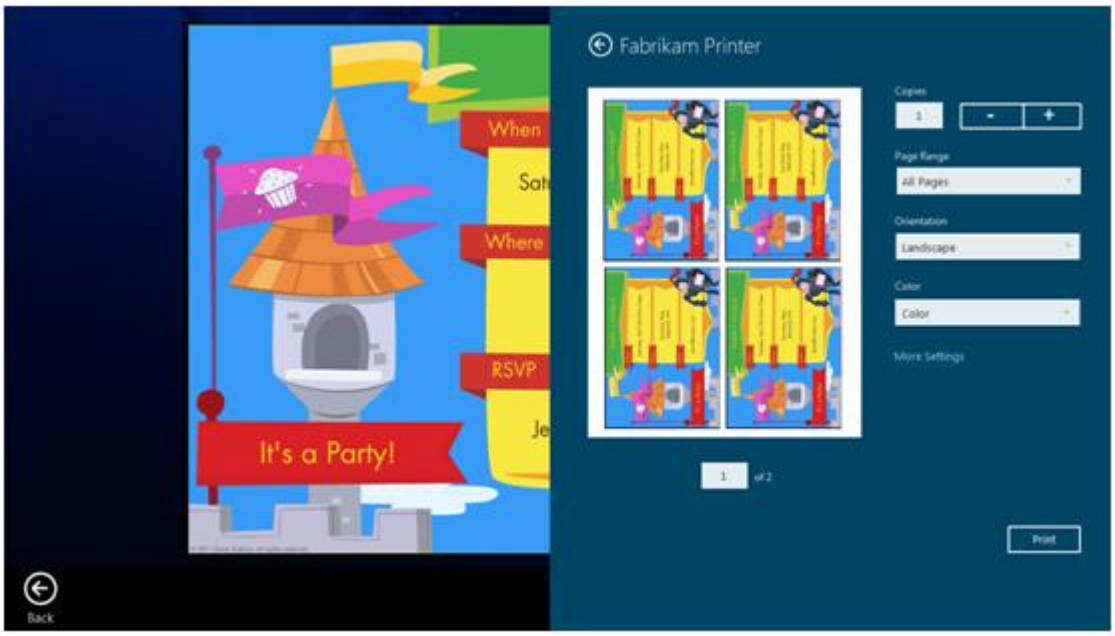
カスタマイズを行っていない場合、これが、Amelia が目にする Windows の既定の印刷設定ウィンドウです。



印刷設定ウィンドウに対するこのカスタマイズとブランド化が行われた Windows ストア デバイス アプリで、Amelia は、ページあたりの写真の枚数を変更できます。

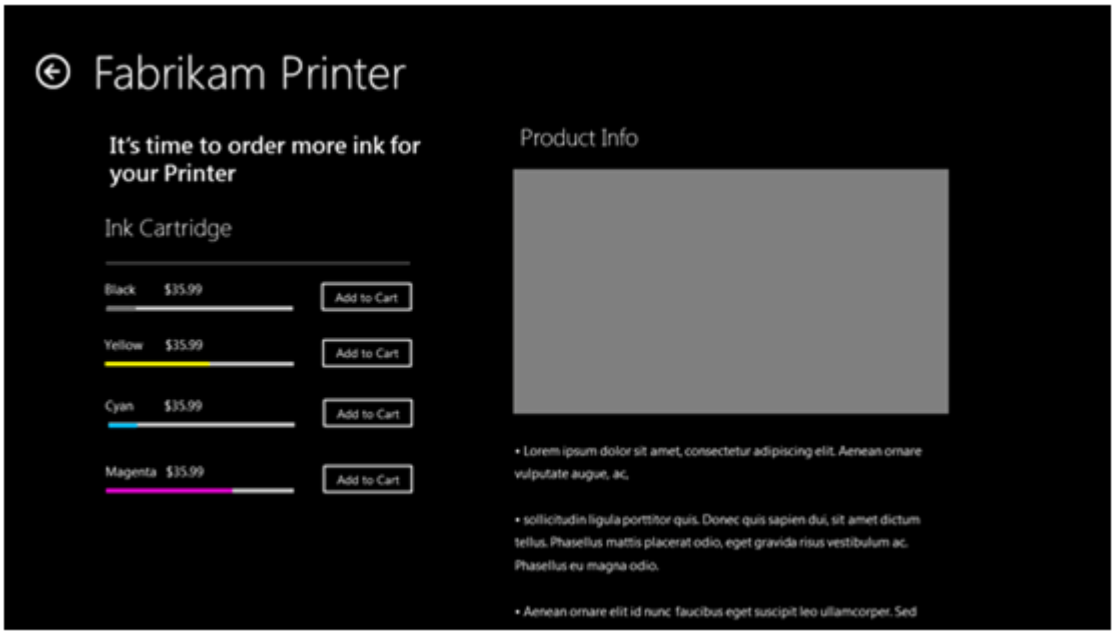


次に、Amelia は **[Back]** ボタンを選びます。 これにより新しい設定または印刷設定が自動的に保存され、印刷ウィンドウに戻ります。

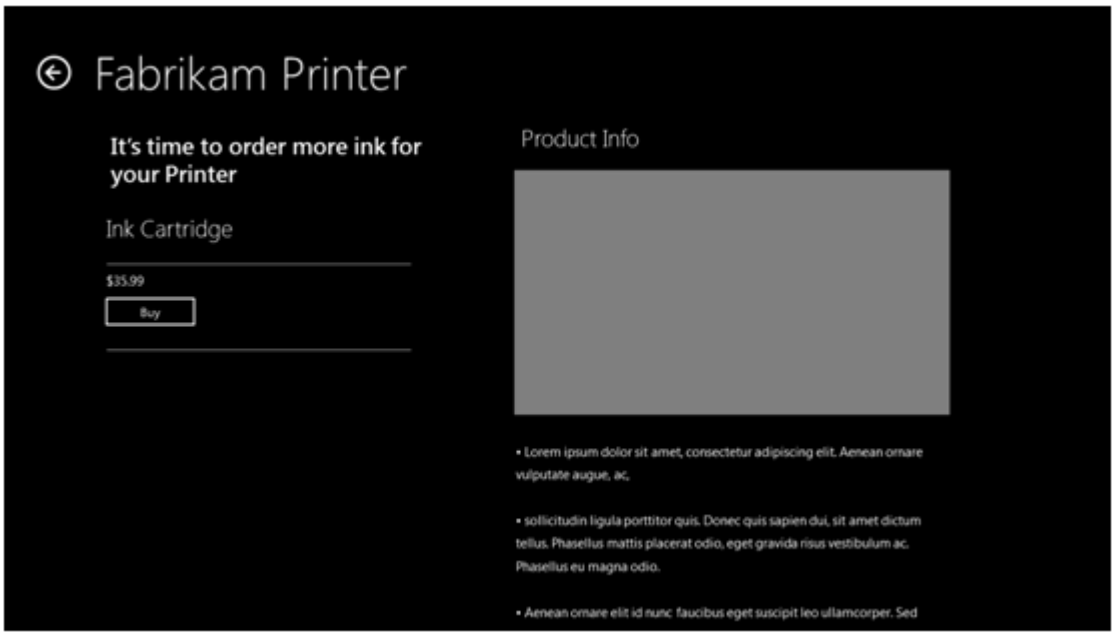


Amelia は、新しい印刷設定を検証した後、**[Print]** を選びます。

Amelia が **[Print]** を選んだとき、使っているアプリが、インク残量が少なくなっていることを示すプリンターからの通知を受け取ります。この通知フライアウトは、トースト通知と呼ばれます。



Amelia がトーストを選ぶ (またはタッチする) と、特定のインク カートリッジのインク残量が少なくなっていることが次の通知ウィンドウに示されます。そこで Amelia は、交換用インク カートリッジを注文するために **[Add to Cart]** を選びます。



Amelia がインク カートリッジをカートに追加すると、次のウィンドウで、注文内容に関する詳しい情報が表示されます。Amelia は、情報を確かめ、**[Buy]** を選びます。

次に、Amelia は、**[Back]** ボタンを選んで (またはタッチして) 印刷ウィンドウに戻ります。Amelia は、このウィンドウで **[Print]** を選んで招待状を印刷できます。

プリンター用 Windows ストア デバイス アプリ作成の必要性

プリンター用 Windows ストア デバイス アプリを使う場合

- 1 ページあたり複数枚の写真の印刷など、高度なデバイス機能を重視する場合。
- デバイスに固有の推奨事項を示します。たとえばデバイス アプリを使って、イメージ管理オプションを表示し、プリンター固有の既定値の設定と保存を行うメソッドを提供できます。

推奨と非推奨

- window.print() を呼び出した後、アプリの印刷ボタンの onClick イベント ハンドラー内からエラー メッセージを調べて処理します。これにより、たとえば使うことができるプリンターがない場合にアプリが印刷要求を中止できます。
- 印刷が失敗した場合は、そのことをユーザーに通知し、可能であれば、エラーの理由を説明します。
- 印刷エクスペリエンスをカスタマイズする場合は、このコードを印刷コンパニオン アプリに分割します。これにより、コードをコンポーネント化でき、テスト プロセスとデバッグ プロセスが容易になります。
- V3 印刷ドライバーを使うように印刷エクスペリエンスをカスタマイズしないでください。
- カスタマイズした印刷 UI で、印刷デバイスのアクセサリを宣伝しないでください。
- Windows ストア デバイス アプリが起動された理由とは無関係の販売品目を表示しないでください。たとえば、ユーザーがインクの残量が少ないことを警告する通知をクリックした場合、その後で販売用の印刷カートリッジを表示することには関連性があります。しかし、この同じシナリオで、印刷コードや写真印刷キットまで販売することは適切ではありません。
- 製品販売を拡大するために、ユーザーを自社の Web サイトにリダイレクトしないでください。
- 印刷設定の設定とは無関係の情報は表示しないでください。たとえば、印刷ヘッドのクリーニング方法、印刷ノズルの位置合わせやテストを行う方法に関する情報は表示しないでください。

近接通信のガイドライン

このトピックでは、近接通信を使ってアプリを接続し、コンテンツを共有する際のベストプラクティスについて説明します。

近接通信は、2 台のデバイスで実行されているアプリの 2 つのインスタンス間で共有されるアプリのエクスペリエンスを実現するための優れた方法です。近接通信が有効なアプリでは、アプリのユーザーが 2 台のデバイスを同時にタップして接続を開始したり、ユーザーがワイヤレス範囲内でアプリを実行している別のデバイスを検索したりすることができます。PC では、ユーザーは Wi-Fi Direct を使って他の PC で実行されているアプリを検索できます。Windows Phone では、ユーザーは Bluetooth を使って他の Windows Phone で実行されているアプリを検索できます。

近接通信を使って通信を行うには、いくつかの方法があります。

- **帯域外セッション:** アウトオブバンド トランスポート (Bluetooth、インフラストラクチャ ネットワーク、または Wi-Fi Direct) でデバイスどうしを接続する **PeerFinder** オブジェクトを使って、セッションを確立できます。 タップの場合の範囲は 3 ～ 4 センチに制限されますが、アウトオブバンド トランスポート オプションの場合の範囲はそれよりも広くなります。 リソースを共有するだけであれば、アプリに近接通信を含める必要はありません。Windows で共有シナリオがサポートされている場合は、共有コントラクトを有効にし、Windows の組み込み機能を使って、タップ ジェスチャでリソースを共有します。

  注 Wi-Fi Direct は Windows Phone ではサポートされていません。

- **ピアの参照:** **PeerFinder.FindAllPeersAsync()** メソッドを使って、セッションを確立できます。このメソッドは、**PeerFinder.Start()** メソッドを呼び出してピア セッションで使用可能であることをアドバタイズしているすべてのリモート ピアを検出します。ピアの参照では、タップ ジェスチャーを使いませんが、代わりに、Wi-Fi Direct を使って、接続を確立するリモート ピアを検出し、接続を確立します。

  注 ピアの参照は、Windows Phone アプリで Bluetooth を使って実行します。このため、Windows Phone で実行されているアプリは電話のピアのみ

を検出でき、コンピューターで実行されているアプリはコンピューターのピアのみを検出できます。

- **メッセージの発行と購読:** ProximityDevice オブジェクトを使って、タップ ジェスチャーを行っているときにメッセージを送受信できます。

アプリが **ConnectAsync** メソッドを呼び出してピアとの接続を作ると、アプリは接続のためのアドバタイズを行わなくなり、アプリが **StreamSocket.Close** メソッドを呼び出してソケット接続を閉じないと **FindAllPeersAsync()** メソッドで見つからなくなります。

ピアが見つかるのは、コンピューターがワイヤレス範囲内にあって、ピア アプリがフォアグラウンドで実行されている場合のみです。ピア アプリがバックグラウンドで実行されている場合、近接通信ではピア接続のためのアドバタイズを行いません。

ConnectAsync メソッドを呼び出してソケット接続を開く場合、コンピューターで開くことができるソケット接続は一度に 1 つのみです。自分のアプリ、または別のアプリが **ConnectAsync** メソッドを呼び出すと、既にあるソケット接続が閉じられます。

デバイス上の各アプリから別のデバイス上のピア アプリに対して開かれた接続がタップ ジェスチャーを使って確立された場合、各アプリでは開かれた接続を 1 つ使うことができます。各デバイスをタップすると、1 つのアプリから複数のデバイス上のピア アプリに対してソケット接続を開くことができます。タップ ジェスチャーを使って接続を作る場合、新しいタップ ジェスチャーで既にある接続は閉じられません。ソケット オブジェクトの **StreamSocket.Close** メソッドを呼び出し、タップ ジェスチャーを使って同じピア デバイス上の同じピア アプリに対して新しい接続を作る必要があります。

推奨と非推奨

- アプリで同じアプリを実行している他のピアを参照する場合は、ピアを連続して参照しないようにします。代わりに、Wi-Fi 範囲内のピアを参照するためのオプションを用意して、ユーザーが操作を開始できるようにします。
- 接続された近接通信エクスペリエンスを開始し、アプリをマルチユーザー モードにする)ときは、必ずユーザーの同意を得る必要があります。たとえば、あるゲームをプレイする 2 人のプレイヤーがいる場合は、プレイヤーが同意を示したうえで

ゲームを一緒にプレイできるようにします。 アプリの起動時にタップが発生した場合は、スタート メニューまたはアプリのロビーで同意を示す機会をユーザーに提供する必要があります。

- ユーザーがアプリをマルチユーザー モードに移行したときは、UI に接続状態を表示する必要があります。接続状態は、次の 3 つのうちのいずれかになります。
  - タップの待機中
  - デバイスに接続中 (進捗状況を表示)
  - デバイスへの接続完了、または接続の失敗
- 接続が中断した場合または設定できない場合は、単一ユーザー モードに戻します。接続が失敗したことを示すメッセージを表示します。
- ユーザーが近接通信エクスペリエンスを簡単に終了できるようにします。
- 同じアプリを実行している他のデバイスを連続して参照しないようにします。代わりに、Wi-Fi 範囲内のピアを参照するためのオプションを用意します。
- 接続に関して定期的な更新 (帯域幅使用状況や速度に関する更新など) を必要とするアプリの場合は、近接通信を使わないでください。

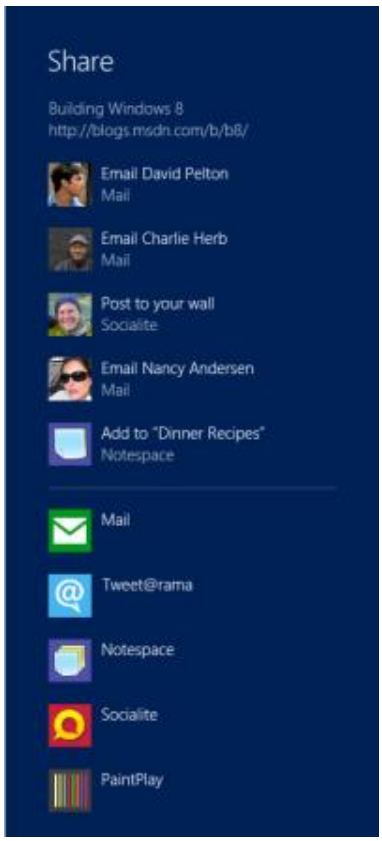
コンテンツの共有のガイドライン

共有コントラクトを使うと、ユーザーはアプリのコンテンツを共有できるだけでなく、インストールされた他のアプリから共有コンテンツを受け取ることもできます。アプリは、共有ソース、共有ターゲット、またはその両方として登録できます。このトピックでは、Windows ストア アプリ間でコンテンツを共有するためのベスト プラクティスについて説明します。

例

ユーザーが画面のエッジ (端) をスワイプして共有チャームをタップすると、共有ペインが開き、ユーザーがコンテンツを共有できるアプリの一覧が表示されます。この一覧には、特定のデータ形式の “共有ターゲット” であるアプリがすべて含まれます。

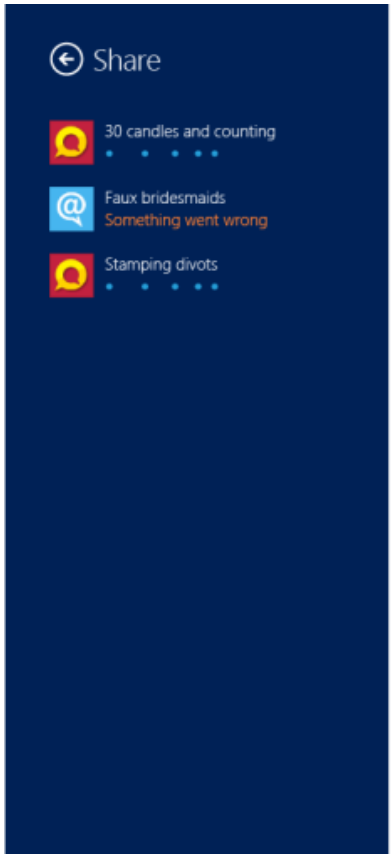
画像の上部にあるリンクは QuickLink と呼ばれ、ユーザーが特定の共有タスクを直接実行するのに使用できます。たとえば、ユーザーがこのコンテンツを共有することを選ぶと、QuickLink が表示され、よく使われる連絡先に URL を電子メールで送ったり、ユーザーのソーシャル メディア アプリのウォールに URL を投稿したり、メモ帳アプリの特定のページに URL を追加できるようになります。「**QuickLink class**」をご覧ください。

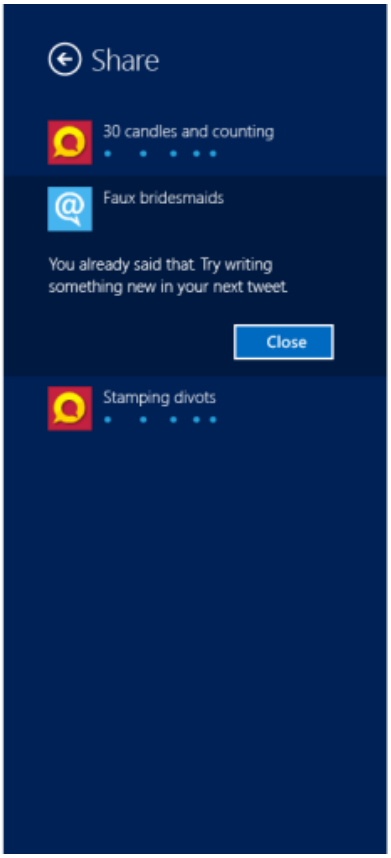


ユーザーが共有ターゲットの一覧から "Tweet@rama" を選んだときに表示される共有ポップアップを次に示します。

時間のかかる共有操作

時間のかかる共有操作を行うための共有ポップアップの例を次に示します。共有に失敗した場合、進行状況不定バーと情報エラー メッセージがターゲット アプリによって表示されます。詳しくは、「プログレス コントロールのガイドライン」をご覧ください。





推奨と非推奨

ソース アプリ

- ユーザーの意図している形式でコンテンツを共有します。たとえば、ユーザーが Web ページの一部だけを共有するように選んだ場合は、その Web ページ全体へのリンクを共有するのではなく、選んだテキストを共有します。
- ユーザーが何を共有しているかを示す説明を提供します。たとえば、ユーザーが Web ページを共有している場合は、ページの URL を示す文章を追加します。画像を共有している場合、画像の説明やタイトルを含めます。
- ユーザーが、表示されているアプリのコンテンツの一部を選んでいる場合は、共有フライアウト (ウィンドウ) が閉じた後、この選択内容を保持します。これは、ユーザーが選択内容を変更する場合や、同じコンテンツを複数のターゲットで共有する場合に役立ちます。
- ダウンロードされたローカル コンテンツのコピーを共有するのではなく、ローカル コンテンツのオンライン版へのリンクを共有します。たとえば、あるニュース サイトが、独自のニュース アプリを作成する一方で、同じ記事を Web サイトでも公開しているとします。ユーザーがソーシャル ネットワーキング サイトと記事を共有する場合、アプリでは、ユーザーが現在表示しているオンライン版の記事へのリンクを共有する必要があります。
- 共有を実行できない場合はユーザーに通知します。特定の共有操作が失敗した場合は、共有フライアウトにメッセージを表示して、問題点を説明し、可能であれば問題の解決策を示します。**DataRequest** オブジェクトでは、このような状況に役立つ **FailWithDisplayText** メソッドがサポートされています。
- アプリ内でデータをコピーする方法をアプリがサポートしている場合は、その同じデータを共有する方法も提供する必要があります。
- プロパティを設定して、ユーザーが共有するコンテンツについての役立つ情報をターゲット アプリに提供します。利用可能なプロパティについて詳しくは、「**DataPackage.DataPackagePropertySet**」をご覧ください。
- アプリが共有をサポートしていない場合は、メッセージを表示しない。アプリが共有コントラクトをサポートしていない場合は、標準メッセージが表示されます。

ターゲット アプリ

- ターゲット アプリと主要なアプリは同じ外観にする。ターゲット アプリの UI は、主要なアプリをよく使うユーザーになじみやすいものにする必要があります。フォント、色、コントロールの一貫性を維持します。
- セットアップとサインインのプロセスが 1 回の操作で実行できる場合は、コンテキストを変更する必要がないように、ユーザーが共有チャームからタスクを完了できるようにしてください。
- 特定のデータ形式についてアプリがソースとターゲットの両方である場合、既定では、ユーザーがそのアプリを使って共有するたびに、共有ターゲットの一覧にそのアプリが表示されます。同じアプリを使ってコンテンツを共有することがユーザーにとって意味を成さない場合、別のターゲット アプリの選択を求めるエラー メッセージが表示されます。
- 共有状態を誤って終了するリンクを削除します。たとえば、同じターゲット アプリ内の他の領域 (ホーム ページなど) に移動するリンクがある場合は、ユーザーが誤って共有状態を終了しないように、それらのリンクを削除するか、隠す必要があります。
- できる限り、プレビューを実際のコンテンツと一致させます。アプリでユーザーが何を共有しているかをプレビューできる場合は、そのプレビューを実際に共有されている内容とできるだけ一致させる必要があります。
- ユーザーの操作を通知します。ユーザーが共有チャームをタップしたとき、または共有 UI を呼び出したときは、ユーザーが共有ペインを閉じる前に、システムがユーザーの操作に対応していることをインライン メッセージなどで知らせてください。これによってユーザーは、共有が無事に開始されたことを確認できます。
- **QuickLink class** を利用します。QuickLink はアプリへのリンクで、特定の一連のユーザー操作 (特定のユーザーに電子メールを送るなど) に合わせてカスタマイズされます。前の「例」のセクションで、利用可能なターゲット アプリの一覧の上に表示されているリンクは、QuickLink です。
- ターゲット アプリでページ間を移動する、戻るボタンを作成しないでください。ユーザーがコンテンツを共有するアプリを選ぶと、利用可能なターゲット アプリの一覧に戻るナビゲーションを支援する、戻るボタンが Windows に自動的に表示

されます。インライン コントロールとエラー メッセージを使用して、ターゲット アプリのナビゲーションをシンプルに保ちます。

- ターゲット アプリでは時間のかかる操作や複雑な操作、または複数のステップから成る対話操作を実行しないようにします。テキストに書式を設定したり、写真の人物にタグを付けたり、データ ソースへの接続などのタスクをセットアップする操作は、共有チャームの外部で行うことが適切です。アプリで複数のステップから成るサインインまたはセットアップ処理が必要な場合、アプリを直接開いてより複雑な操作を完了するようにユーザーに伝えます。 サインインまたはサインアップの対話操作で Web 認証ブローカーを使うことを計画している場合は、「Web 認証ブローカー」をご覧ください。

その他の使い方のガイダンス

ターゲット アプリのデバッグ

共有ターゲット アプリは共有チャームからのみ起動することができ、アプリの外側をクリックすると閉じられます。つまり、デバッガー内でクリックするとターゲット アプリが閉じるので、共有ターゲットに対しては多くのローカル デバッグ シナリオは実用的ではありません。共有ターゲット アプリをデバッグするには、通常、シミュレーターまたはリモート デバッグを使う必要があります。仮想または物理的に、別のマシンで操作するためです。

ローカルでデバッグできる共有ターゲットのシナリオの場合、デバッガーがデタッチするので、Visual Studio から起動した後はターゲット アプリを閉じることはできないことを忘れないでください。代わりに、全画面でアプリを開いたままにしておき、スタート画面に移動してから、共有シナリオを実行します。

デバッグ方法

共有ターゲット アプリは、起動後ほんの一瞬で、通常はデバッガーを実行中のプロセスにアタッチできる前に、アクティブ化されます。アクティブ化ハンドラーのコードをデバッグするには、ローカル マシンで直接デバッグするか、またはシミュレーターを使うことをお勧めします。

ユーザーが別のアプリに移動して実行できる長時間の共有を実装した場合、アプリの UI が閉じた後でエラーが発生する可能性があります。この場合、ソース アプリが画面から消えても、デバッガーをターゲット アプリにアタッチしたままにできます。デバッガーがアタッチされた状態で、例外をキャッチし、ブレークポイントで一時停止して、画面でのアプリ UI なしでコードを確認できます。

一般的なデバッグの問題

- ターゲット アプリは処理されない例外により直ちに終了され、エラー メッセージで置き換えられます。ターゲット アプリは、無効な入力データなどのユーザーから発生する予想されるエラーを適切に処理し、ユーザーに報告する必要があります。
- ターゲット アプリでアクティブ化イベントへの応答に時間がかかりすぎる場合、システムはアプリが応答しないことを選んだと判断し、エラーを表示します。可能な限り、データの処理はアクティブ化ハンドラーの外に移動する必要があり、通常は **ShareOperation** オブジェクトを格納し、非同期に処理することで行います。
- API の共有への呼び出しは、呼び出しの回数が多すぎる場合や順番が間違っている場合に、例外をスローできます。長時間の共有を実装する場合、必ず次の順番で共有メソッドを呼び出し、単一のメソッドを 2 回連続で呼ぶことがないようにしてください。いつでも **ReportError** または **ReportCompleted** を呼び出して共有操作を完了できます。

ファイル、データ、接続のガイドライン

このセクションでは、アプリからファイルやデータにアクセスする方法と、ユーザーを Web やユーザーのアカウントに接続する方法について説明します。

ユーザーに対し、アプリからフォルダー、ファイル、データへのアクセスを許可できます。ユーザーがアカウントから、または OneDrive などのクラウド サービスにあるデータにアクセスできるように、ユーザーにサインイン エクスペリエンスを提供できます。またアプリにアプリケーション データ ローミングを含めると、デバイス間でのシームレスなエクスペリエンスが実現します。

このセクションの内容

トピック	説明
接続の使用状況データ	接続されたアプリで Windows ランタイム Network Information API を使うときは、以下の推奨事項を考慮してください。
カスタム データ形式	ユーザーは、さまざまな情報をオンラインで共有します。成功するアプリを作成するには、ユーザーが共有する可能性が高い情報の種類を分析し、その情報を受信側のアプリが正しく処理できるようにパッケージ化します。ユーザーが共有する情報は、多くの場合、Windows でサポートされている 6 つの標準形式のいずれかに当てはまります。しかし、より的を絞ったデータ型を使うことによってユーザー エクスペリエンスを向上させることができる場合もあります。そのような場合は、アプリでカスタム データ形式をサポートすることができます。
ファイルの種類と URI	Windows ストア アプリと、アプリがサポートするファイルの種類やプロトコルの間の関係を理解すると、より一貫性があり、洗練されたエクスペリエンスをユーザーに提供できます。
ログイン	多くの Windows ストア アプリは、ユーザーがログインすると、カスタマイズされたエクスペリエンスや初回用エクスペリエンスを提供します。アプリを活用するのに、登録したアカウントへのログインが必須である場合もあります。一方で、どのユーザーにもリッチな基本エクスペリエンスを提供し、ユーザーがログインしたら拡張機能を有効にするアプリもあります。

アプリ データのローミング	ローミング アプリ データが含まれるように Windows ストア アプリを設計するときは、次のガイドラインに従ってください。
OneDrive	Microsoft OneDrive にあるユーザーのファイル、ドキュメント、画像、動画、フォルダー、アルバム、コメントを操作する Windows ストア アプリを設計する場合は、次のガイドラインに従ってください。
シングル サインインと接続されているアカウント	Microsoft アカウントを持っているユーザーに対して認証されたエクスペリエンスを提供する、Windows ストア アプリのガイドラインを説明します。
サムネイル	このガイドラインでは、サムネイル イメージを使って、Windows ストア アプリでファイルをプレビューする方法について説明します。
ユーザー名とアカウントの画像	Windows 8 の場合、アプリで現在のユーザーの名前と画像 (アカウントの画像) を取得し、それらを使って、ユーザーを識別したうえで、ユーザーに合わせたエクスペリエンスを作成できます。

接続の使用状況データのガイドライン

接続されたアプリで Windows ランタイム Network Information API を使うときは、以下の推奨事項を考慮してください。

ネットワーク コストの種類に応じたアプリ動作の変更

デバイスで新しいネットワークが検出されると、Windows 8 により新しい接続オプションが提供されますが、それまでの接続から新しいネットワークに毎回スムーズに移行できるとは限りません。Web に接続する Windows ストア アプリは、Network Information API を使って、データを送受信するネットワークのコスト情報とステータス変更イベントを取得する必要があります。

動作を適切に変更するには、各接続に対して指定されている **NetworkCostType** 値を使います。

ネットワーク コスト タイプ	推奨されるアプリの動作
制限なし	• 自由にネットワーク接続を使います。
Variable (データの上限が迫っているとき)	• 無制限のネットワークを使うことができるまで、優先度の低い操作を遅らせるか、スケジュールを設定します。 • ムービーや動画などのコンテンツをユーザーにストリーミングしている場合は、ビット レートを低くします。たとえば、高精細 (HD) 画質のビデオをストリーミングしている場合、従量制課金接続では標準画質に切り替えます。 • 使用帯域幅を減らします。たとえば、電子メールを受信する場合、ヘッダーのみをダウンロードするモードに切り替えます。 • ネットワークの使用頻度を減らします。たとえば、ニュース フィードをダウンロードしたり、Web サイトのコンテンツを最新の情報に更新したり、Web 通知を取得したりするときのポーリング操作の回数を減らします。

	• 高精細 (HD) 画質のビデオのストリーミング、電子メール全体の取得、優先度の低い更新プログラムのダウンロードなどは、ユーザーが明示的に選べるようにします。このような操作を既定で行うことは避けてください。 • ネットワークを使う前に、ユーザーにはっきりと許可を求めます。
Unknown (不明)	• ネットワーク コスト タイプが不明である場合、上限のないネットワークとして扱います。

Web への信頼性の高い接続の維持

ネットワーク空間におけるアプリの機敏性を維持する最も基本的な方法の 1 つは、Web と情報をやり取りする際の通信品質を一定のレベルに保つことです。そのためには、接続プロファイルから得られた情報と、以後伝えられるネットワーク ステータス変更通知を利用し、現在の要件を満たした利用可能なネットワークを識別する必要があります。

すべての Windows ストア アプリは、Web 接続をサポートするために、次のことを実行する必要があります。

1. **GetInternetConnectionProfile** を呼び出して、インターネットへの接続コストをチェックします。
2. 接続のネットワーク ステータス変更通知を受け取るための登録を行います。
3. 接続のネットワーク操作を初期化します。
4. ネットワーク ステータス変更通知は、利用できるコストや接続のオプションが変わっている可能性を示します。アプリは以下を実行する必要があります。
   - インターネットへの接続コストをチェックします。コスト体系が変わっていた場合 (無料から有料か、有料から無料) は、ネットワーク操作を再試行します。Windows 8 では、最も低コストのネットワークが自動的に選ばれます。
   - インターネット接続のコスト体系は変わっていないが、コストに関連した通知を受け取った場合 (データ使用量が上限の 80% を超えた、変動制、ローミングなど)、上記の **NetworkCostType** の表に従って動作を調整します。
5. 接続が切断されたことがエラーで示された場合は、アプリで以下のことを実行します。

- 利用可能な別のネットワークを使ってインターネットへの接続コストをチェックします。上記の **NetworkCostType** の表に記載したガイドラインに従ってください。
- 操作を再試行します。失敗した場合は、**NetworkStatusChange** 通知を待ちます。

接続アプリのデバッグとトラブルシューティング

ネットワークの問題は、アプリがハングしたり、クラッシュしたりする原因となります。対処不可能なダイアログ ボックスや紛らわしいエラー メッセージがユーザーに表示されることもあります。こうしたエラーは、ネットワーク スタックのどこで発生してもおかしくないので、デバッグが困難な場合があります。

ネットワークを (ソケットで) 直接的に、または (ネットワークを使う API で) 間接的に使うすべての Windows ストア アプリに関係します。通常はエラーの発生条件を開発者に代わりオペレーティング システムが自動的に処理し、それができない場合、アプリでエラーに対処できるのが理想です。

すべての接続 Windows ストア アプリに求められる対応を次に示します。

- ネットワーク エラーが発生したら、状況に応じて操作を再試行します。たとえば、認証に失敗した場合には操作を再試行しません。これに対し、通信中のネットワークがつながらなくなった場合は、別のネットワークが利用できる可能性があるので操作を再試行します。操作を再試行するだけで、多くのエラーは解消されます。再試行の際は、「ネットワーク ステータスの変化への対応」に記載したガイドラインに従ってください。
- 必ず非同期 API を使うようにし、ブロック呼び出しが UI スレッドに存在しないようにします。つまり、ネットワーク操作で完了までに時間がかかったりエラーが発生したとしても、アプリがハングしないようにします。Windows ランタイムの非同期的な特性を、同期的な動作で押さえつけてしまうことのないように注意してください。
- ネットワークの切断と再接続、中断と再開、ネットワークの切り替えなどの操作を行いながら、さまざまなネットワーク環境でアプリをテストします。
- アプリケーションをテストしていて、一見してわからないようなエラーが見つかった場合は、ETW トレースを有効にします。

カスタム データ形式の作成のガイドライン

ユーザーは、さまざまな情報をオンラインで共有します。成功するアプリを作成するには、ユーザーが共有する可能性が高い情報の種類を分析し、その情報を受信側のアプリが正しく処理できるようにパッケージ化します。ユーザーが共有する情報は、多くの場合、Windows でサポートされている 6 つの標準形式のいずれかに当てはまります。しかし、より的を絞ったデータ型を使うことによってユーザー エクスペリエンスを向上させることができる場合もあります。そのような場合は、アプリでカスタム データ形式をサポートすることができます。

例

ここでは、カスタム形式の作成についての考え方をよりわかりやすく示すために、次の例について検討します。

Fabrikam という架空の会社の開発者が、オンラインに保存されているファイルを共有するためのアプリを作成します。1 つのオプションとしてストリーム方式の StorageItems を使う方法が考えられますが、ターゲット アプリでファイルを読むためにはファイルのダウンロードが必要になるため、時間がかかり、非効率です。そのため、それらのファイルの種類で使うためのカスタム形式を作成することにしました。

まず、新しい形式の定義について検討します。この場合、新しい形式は、オンラインに保存されているファイルの種類 (ドキュメント、画像など) のコレクションです。これらのファイルはローカル コンピューターではなく Web にあるため、形式の名前は WebFileItems にすることにしました。

次に、その形式の詳細について決定する必要があります。次のように決定されました。

- この形式は、URI を表す InspectableArray を含む **IPropertyValue** で構成されます。
- この形式には、少なくとも 1 つの項目が含まれている必要がありますが、含めることのできる項目の数に制限はありません。
- 任意の有効な URI を使うことができます。
- ソース アプリケーションの境界の外からはアクセスできない URI (認証される URI など) は推奨されません。

これで、カスタム形式を作成して使用するための十分な情報が集まりました。

アプリでカスタム形式を使うかどうか

Windows 8 の共有機能では、次の 6 つの標準データ形式がサポートされています。

- Text
- HTML
- Bitmap
- StorageItems
- URI
- RTF

これらの形式は汎用性が高いため、共有コンテンツの共有と受信をアプリですばやく簡単にサポートすることができます。その一方で、受信側のアプリに対してデータの豊富な説明が必ずしも提供されないという欠点もあります。たとえば、住所を表す次のような文字列があったとします。

1234 Main Street, New York, NY 98208

アプリでこの文字列を共有するには、**DataPackage.setText** を使います。しかし、受信側のアプリでは、この文字列が何を表すのか正確にはわからないため、このデータを使ってできることは限られます。カスタム データ形式を使うと、共有されるデータを、http://schema.org/Place 形式を使った "場所" としてソース アプリで定義することができます。これにより、受信側のアプリに追加情報が提供されて、ユーザーの期待どおりに情報を処理できるようになります。既にあるスキーマ形式を使うと、定義されている形式のより大きなデータベースにアプリをフックできます。

カスタム形式は、データをより効率的に共有できるようにするために使うこともできます。たとえば、画像のコレクションを Microsoft OneDrive に保存しているユーザーが、その一部をソーシャル ネットワークで共有することにしたとします。このシナリオは、次のような理由から、標準形式を使って実現するには問題があります。

- Uniform Resource Identifier (URI) 形式は、一度に 1 つの項目しか共有できません。
- OneDrive では、コレクションはストリーム方式の StorageItems として共有されます。このシナリオで StorageItems を使うには、アプリで各画像をダウンロードしてから共有する必要があります。
- Text と HTML ではリンクの一覧を提供できますが、リンクの意味は失われます。したがって、受信側のアプリには、それらのリンクが、ユーザーが共有しようとしている画像を表すことはわかりません。

既に存在するスキーマ形式またはカスタム形式を使ってこれらの画像を共有することには、次の 2 つの大きな利点があります。

- すべての画像をローカルにダウンロードする代わりに URI のコレクションを作ることができるため、アプリでよりすばやく画像を共有できます。
- 受信側のアプリで、それらの URI が画像を表すことを認識し、それに応じた処理を行うことができます。

推奨と非推奨

カスタム形式の定義

アプリでカスタム形式を定義するメリットがあることがわかったら、次の点について検討する必要があります。

- 標準データ形式について理解していることを確認します。理解していないと、無駄にカスタム形式を作成することになります。自分のシナリオに合う形式が既にあるかどうかを http://www.schema.org で確認する必要があります。自分のカスタム形式をカバーする既に存在する形式が別のアプリで使われている可能性があります。その場合は、より多くの対象ユーザーが、自分の考えるエンド ツー エンドのシナリオを実現できることになります。
- 実現するエクスペリエンスについて検討します。ユーザーが求めている操作と、その操作をサポートするのに最適なデータ形式について考えることが重要です。
- カスタム形式の定義を他のアプリ開発者が使用できるようにします。

- 形式には、内容に合った名前を付けるようにします。たとえば、UriCollection は、あらゆる種類の URI が含まれていることを表し、WebImageCollection は、オンライン イメージを指す URI のみが含まれていることを表します。
- その形式の意味について慎重に検討します。その形式が何を表し、どのように使用されるのかを明らかにします。
- その形式の構造について吟味します。複数の項目やシリアル化をサポートするかどうかや、どのような制限があるのかについてよく検討します。
- いったん公開したカスタム形式は変更しないでください。API のようなものと考えて、要素が追加されたり廃止されたりすることがあっても、下位互換性と長期的なサポートが確保されるようにします。
- 形式の組み合わせに依存しないようにします。たとえば、一方の形式が見つかったらもう一方の形式を探すなどの動作をアプリに期待しないようにしてください。形式はそれぞれ自己完結している必要があります。

アプリへのカスタム形式の追加

カスタム形式を定義した後にその形式をアプリに追加するときは以下の推奨事項に従ってください。

- その形式を他のアプリでテストします。ソース アプリからターゲット アプリにデータが正しく処理されることを確認します。
- その形式の当初の目的を見失わないようにします。他の用途に使用しないようにしてください。
- ソース アプリを作成する場合、標準の形式も少なくとも 1 つは提供するようにします。そうすると、そのカスタム形式をサポートしていないアプリでもデータを共有できるようになります。理想的なエクスペリエンスは得られないかもしれませんが、特定のアプリでしかデータを共有できなくなるよりはましです。さらに、他のアプリでこの形式を利用できるように、形式に関するドキュメントをオンラインで提供します。
- ターゲット アプリを作成する場合、標準の形式を少なくとも 1 つサポートすることを検討します。そうすれば、ソース アプリで目的のカスタム形式が使われていなくてもデータを受信できます。

- 他のアプリの特定のカスタム形式 (特に、他社の形式) を利用する場合は、その形式に関するドキュメントがオンラインで一般に公開されているかどうかをもう一度確認する必要があります。ドキュメント化されていない形式は予告なく変更される可能性が高く、不整合が生じたり、アプリが失敗したりする原因となります。

その他の使い方のガイダンス

データの選択

カスタム形式を定義するときに行う最も重要な決定の 1 つは、ソース アプリケーションとターゲット アプリケーションの間の転送に使う WinRT データ型です。**DataPackage** クラスでは、カスタム形式用に以下のデータ型がサポートされています。

- **IPropertyValue** の任意のスカラー型 (integer、string、**DateTime** など)
- **IRandomAccessStream**
- **IRandomAccessStreamReference**
- **IUri**
- IStorageItem
- 上記のいずれかの項目の異種コレクション

カスタム形式を定義するときには、そのデータに適した型を選択します。その形式のすべてのコンシューマーと受信側が (他の選択肢があったとしても) 同じデータ型を使うことがとても重要です。そうしないと、データ型の不一致によるエラーがターゲット アプリケーションで予期せず発生する可能性があります。

カスタム形式のデータ型に string を選ぶと、ターゲット側で **GetTextAsync** 関数の **GetTextAsync(formatId)** 形式を使ってデータを取得できるようになります。この関数では、データ型の確認が行われます。その他のデータ型を使う場合は、**GetDataAsync** を使う必要があります。この場合は、データ型の不一致に対する保護が必要になります。 たとえば、ソース アプリケーションから 1 つの URI が提供された場合に、ターゲットが URI のコレクションとして取得しようとすると、不一致が生じます。こうした競合を防ぐには、次のようなコードを追加します。

DataPackage の設定

```
var uris = new Array();
uris[0] = new Windows.Foundation.Uri("http://www.msn.com");
uris[1] = new Windows.Foundation.Uri("http://www.microsoft.com");
var dp = new Windows.ApplicationModel.DataTransfer.DataPackage();
dp.setData("UriCollection", uris);
```

```
System.Uri[] uris = new System.Uri[2];
uris[0] = new System.Uri("http://www.msn.com");
uris[1] = new System.Uri("http://www.microsoft.com");
DataPackage dp = new DataPackage();
dp.SetData("UriCollection", uris);
```

データの取得

```
if (dpView.contains("UriCollection")) {
    dpView.getDataAsync("UriCollection").done(function(uris) {
        // Array.isArray doesn't work – uris is projected from InspectableArray
        if (uris.toString() === "[object ObjectArray]") {
            var validUriArray = true;
            for (var i = 0; (true === validUriArray) && (i < uris.length); i++) {
                validUriArray = (uris[i] instanceof Windows.Foundation.Uri);
            }
            if (validUriArray) {
                // Type validated data
            }
        }
    }
}
```

```
if (dpView.Contains("UriCollection"))
{
    System.Uri[] retUris = await dpView.GetDataAsync("UriCollection") as System.Uri[];
    if (retUris != null)
    {
      // Retrieved Uri collection from DataPackageView
    }
}
```

ファイルの種類と URI のガイドライン

Windows 8 では、アプリとアプリがサポートするファイルの種類の間の関係は以前のバージョンの Windows とは異なります。これらの違いを理解すると、より一貫性があり、洗練されたエクスペリエンスをユーザーに提供できます。

推奨と非推奨

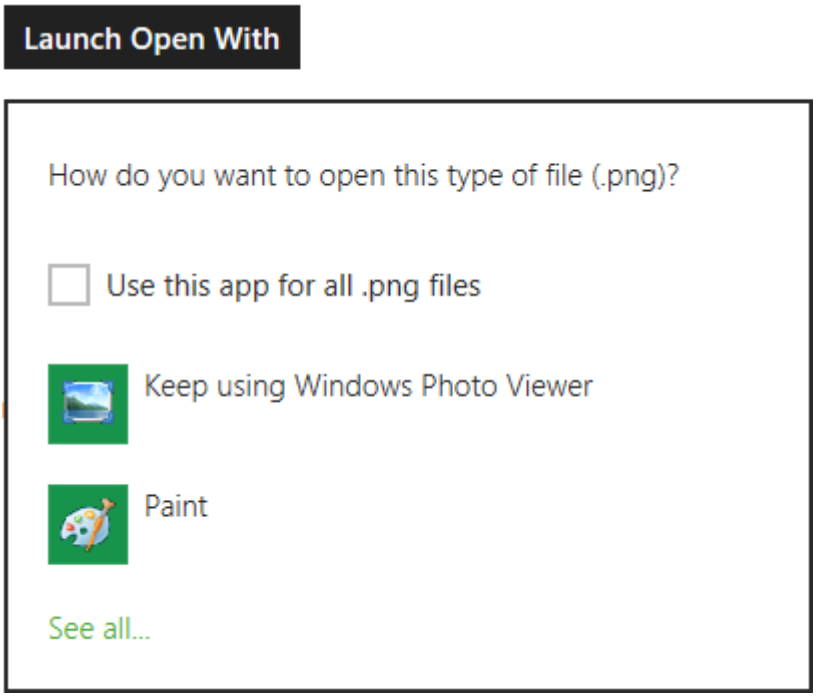
- フライアウトは、呼び出した位置の近くに配置します。

その他の使い方のガイダンス

Windows ストア アプリのガイドライン

ファイルまたは URI を開くときに、既定で使用するアプリを選ぶために、ユーザーが **[プログラムから開く]** の一覧を使うことが必要になる場合があります。Windows 8 ではこの一覧をフライアウトで実装しています。**[プログラムから開く]**フライアウトのコンテンツはカスタマイズできませんが、アプリ内の位置は制御できます。このガイドラインに従って、呼び出した位置のできるだけ近くにフライアウトを配置してください。

フライアウトの望ましい使用方法の例を次に示します。呼び出したボタンのすぐ横にフライアウトが配置されていることを確認してください。



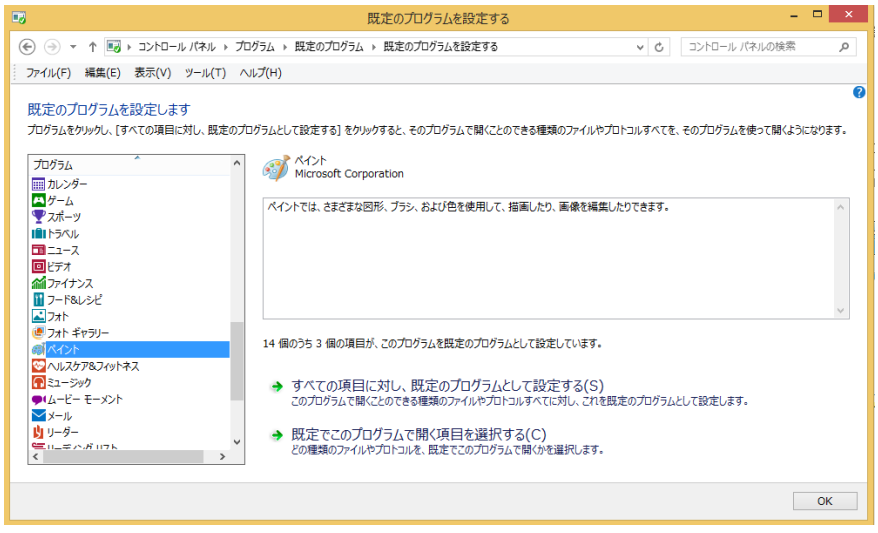
ファイルや URI を適切と思われるもの (通常はサムネイルやハイパーリンクで表示される) に渡すことができます。これらのアイテムのプライマリ動作は **Open** になります。この動作はファイルまたは URI の既定のハンドラーを呼び出し、場合によっては **[プログラムから開く]** フライアウトが表示される結果になります (フライアウトが表示される場合があることを想定して適切に配置することをお勧めします)。

ファイルまたは URI の任意のセカンダリ動作 (Save As や Download など) をアプリに実装する場合は、**[プログラムから開く]** フライアウトでユーザーに別のアプリを選択させることを考えてください。

Windows ストア アプリでは、ファイルの種類や URI に対する既定のアプリの設定、変更、照会はできないため、アプリにこの機能を追加しないでください。

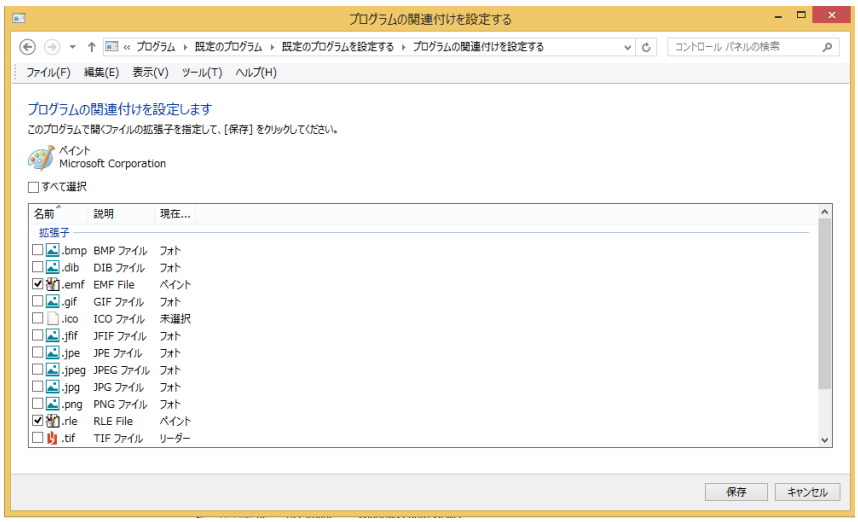
デスクトップ アプリのガイドライン

Windows 8 では、ファイルの種類や URI スキーマ名に対する既定のハンドラーの設定、変更、照会をアプリができなくなりました。アプリの Windows 8 デスクトップ アプリ バージョンを開発するときは、これらの機能に関連付けられたユーザー インターフェイス要素をすべて削除することをお勧めします。代わりに、コントロール パネルの **[既定のプログラムを設定する]** にリンクすることをお勧めします。 **[既定のプログラムを設定する]** の UI を次に示します。

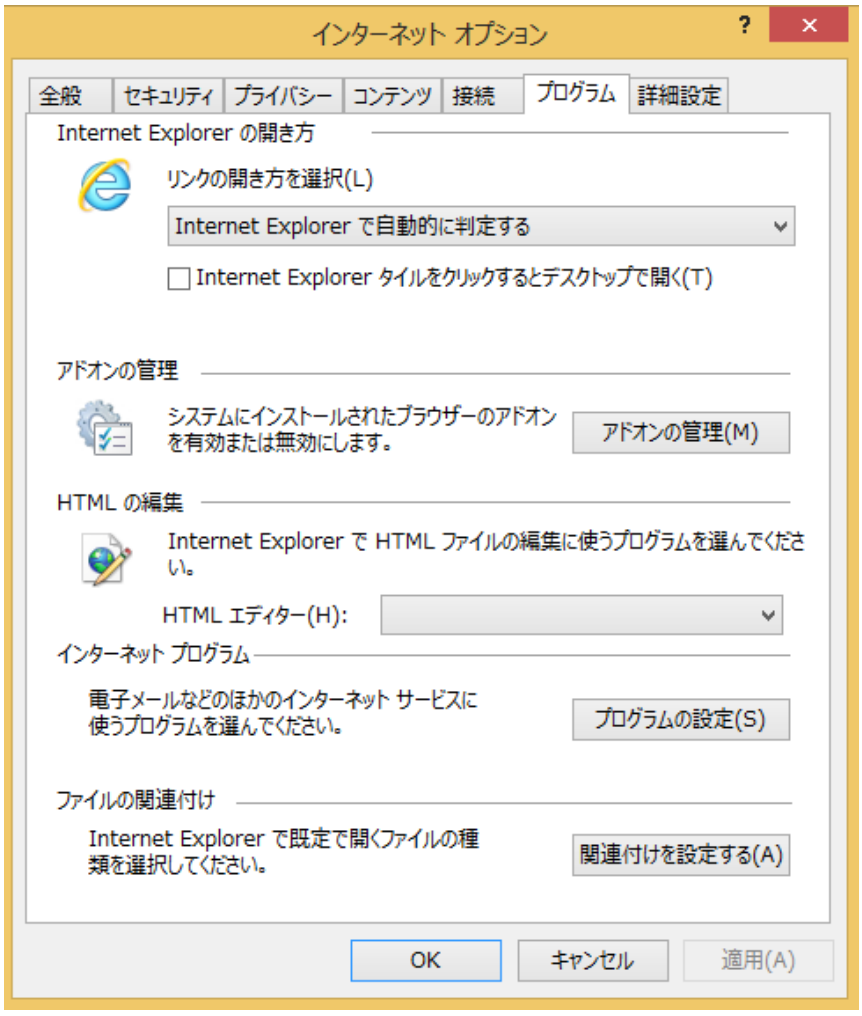


この画面からユーザーは **[すべての項目に対し、既定のプログラムとして設定する]** を選択でき、アプリが開くことができるすべてのファイルの種類と URI を、既定でそのアプリが開くようにできます。また、**[既定でこのプログラムで開く項目を選択する]** を選択して、次

に示す [プログラムの関連付けを設定する] の UI で特定のファイルの種類と URI を選ぶこともできます。



次に、ユーザーをコントロール パネルの **[既定のプログラムを設定する]** にアクセスさせる方法についての Microsoft Internet Explorer の例を示します。



ログインのガイドライン

多くの Windows ストア アプリは、ユーザーがログインすると、カスタマイズされたエクスペリエンスや初回用エクスペリエンスを提供します。アプリを活用するのに、登録したアカウントへのログインが必須である場合もあります。一方で、どのユーザーにもリッチな基本エクスペリエンスを提供し、ユーザーがログインしたら拡張機能を有効にするアプリもあります。

推奨と非推奨

ログイン設定

- アプリの中で、アプリにログインし、アカウントを作り、アカウント設定を管理する手段を提供する場合は、ユーザーがエッジ (端) からスワイプして、設定フライアウトでログイン設定を変更できるようにすることをお勧めします。このような設計にすることにより、ユーザーはアプリ内のワークフローのどこにいても、わかりやすく簡単な方法でアクセスできます。またこの設計では、アプリのキャンバスをログイン関連の UI で占拠しないため、場所を広く使うことができます。

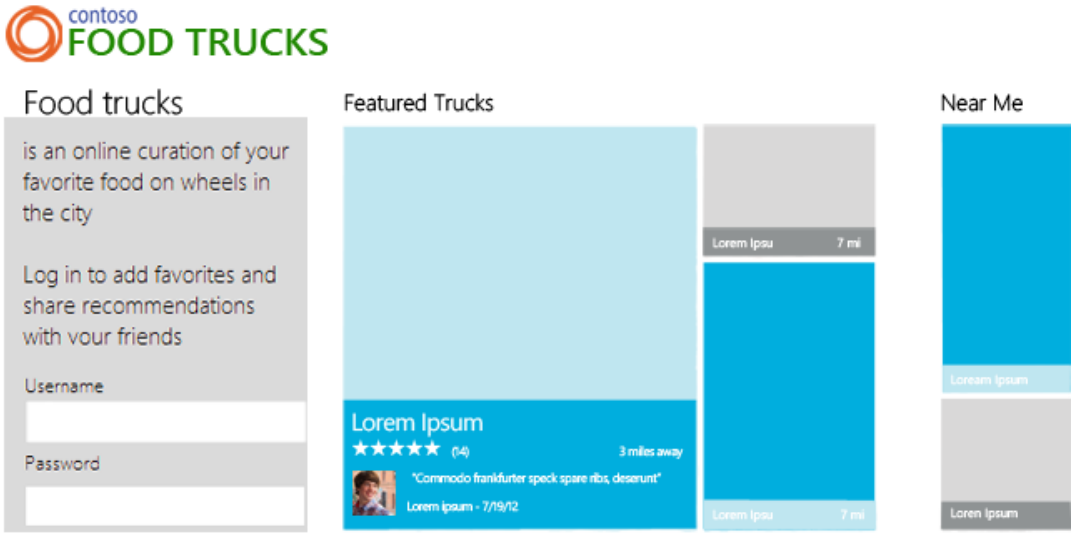
ログインが必須の場合

- アプリにおいて、ユーザーのログイン、またはアプリの初回実行時のアカウント作成を必須とする場合は、アプリの最初の画面でログイン UI を目立つように強調します。ユーザーがログインしたら、画面上にログイン UI を表示する必要はありません。
- ユーザーは設定チャームを使ってログアウトします。

ログインが推奨の場合

- ログイン UI をアプリのキャンバス上に配置する必要がある場合は、コントロールをコンテンツ内にインラインで提供します。このような設計にすると、ユーザーはアプリの初回起動時にランディング ページでログイン オプションを必ず目にすることになりますが、ログイン UI がエクスペリエンス全体を邪魔することはありません。

- ログイン UI を ListView コントロールの最初のセクションとしてアプリのランディング ページに配置します。ユーザーがアプリのコンテンツやビューを閲覧する際、ログオン UI はスクロールによって表示されなくなります。ただし、アプリには存在し続けます。
- 常にユーザーが設定フライアウトでログイン UI を見つけられるようにします。

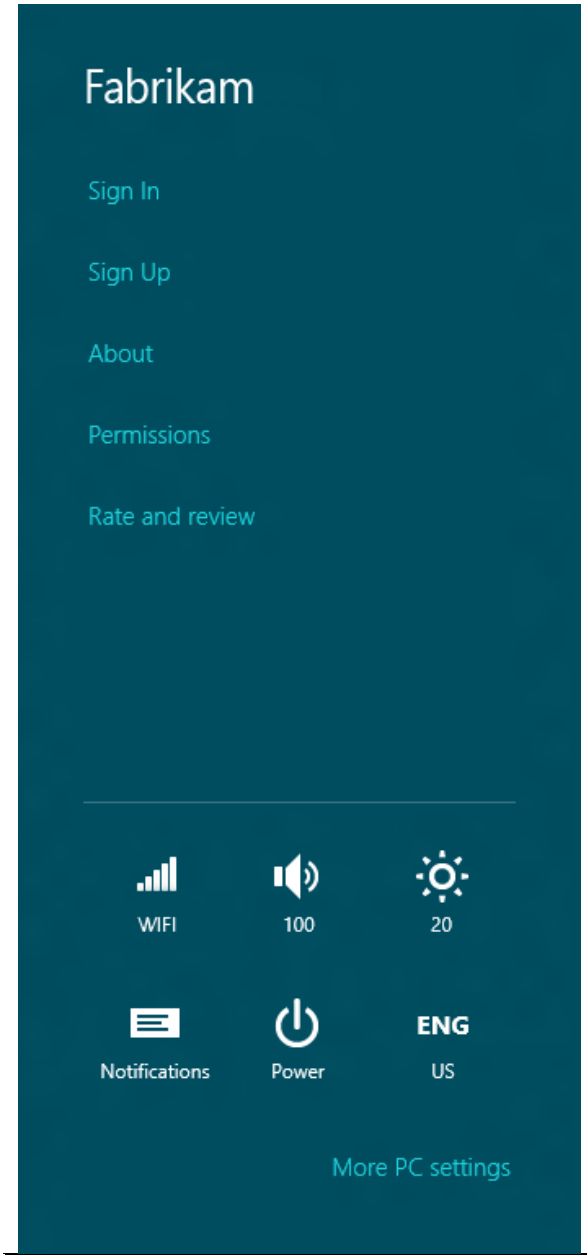


ログイン UI が ListView コントロールの最初のセクションとしてホストされているアプリ

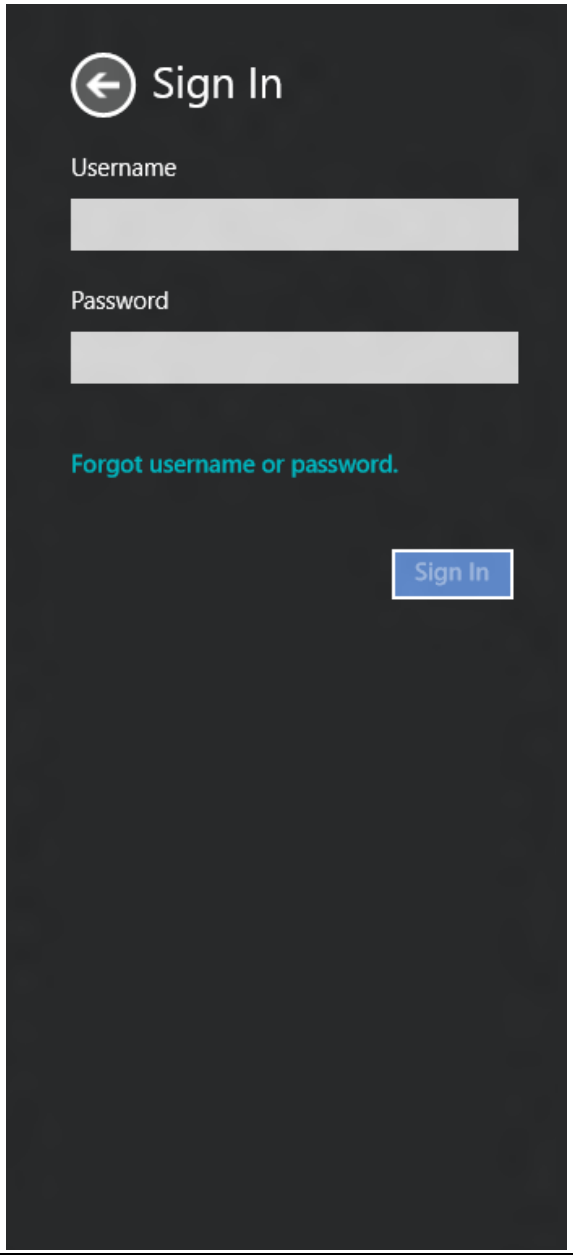
ログインが任意の場合

- アプリへのログインが任意である場合は、ログイン UI を設定フライアウトに配置します。こうすると、ログイン UI のためにアプリのコンテンツからユーザーの注意がそれることや、キャンバス上のスペースが狭くなることがありません。アプリの中には、ユーザーにログインを求めることなく優れた価値を提供するものもあります。たとえば、ニュース アプリでは多くの読者が関心を持つニュース記事の冒頭部分を提供することがあります。ユーザーは、ログインしなくてもアプリから大きな価値を得られます。
- アプリの一部のコンテンツだけに専用のログイン UI が必要な場合は、ユーザーにログインが必要なことを示すために、コンテキストに応じてログイン ボタンをページに表示します。このログイン ボタンで、ログイン UI を提供する設定フライア

ウトを起動します。たとえば、ニュース アプリでユーザーがニュース記事にコメントを投稿するにはログインが必要な場合などです。



設定チャーム フライアウト ログイン



設定フライアウト ログイン

ログアウト UI

- ログアウト UI は設定フライアウトに配置します。アプリに一度ログインしたユーザーは、アプリがそのユーザー向けに有意義なコンテンツを提供している限り、ほとんどログアウトすることはありません。必要に応じて、見慣れた場所でログアウトできるようにします。

ログイン後のアプリのカスタマイズ

- アプリへのログイン後のエクスペリエンスを、アプリをカスタマイズし、接続を持続させるコンテンツを使ってデザインします。
- ユーザーがログインしたら、汎用的なコンテンツを表示するのではなく、そのユーザーの設定に基づいてコンテンツを更新します。各アプリで個人向けのコンテンツを独自の方法で表示し、提供すれば、Windows ストア アプリのユーザー エクスペリエンスを向上できます。
- アプリのキャンバス上に ID を表示する UI を常に配置しておくことは避けます。ユーザーのアプリへのログインと、Windows へのログインは一致しないことがあります。たとえば、自分のものではないノート PC やタブレットを使うユーザーが、ソーシャル ネットワークにログインする場合です。スタート画面とアプリで ID が異なる場合、ユーザーにとってメリットよりも混乱する可能性の方が大きくなります。

アプリ データのローミングのガイドライン

ローミング **ApplicationData** API を使ってアプリのデータを保存すると、Windows によって、このデータはクラウドにレプリケートされ、アプリがインストールされているその他のすべてのユーザー デバイスに同期されます。ローミング アプリ データが含まれるように Windows ストア アプリを設計するときは、次のガイドラインに従ってください。

アプリでローミング データを使うかどうか

ローミング データを使ってユーザーの設定、基本設定、セッション情報を保存し、複数のデバイス間で統一感のあるアプリのエクスペリエンスを提供します。ローミング データがユーザーの Microsoft アカウントに関連付けられていることに注意してください。ローミング データが同期されるのは、ユーザーが同じ Microsoft アカウントを使ってデバイスにログインし、複数のデバイスにアプリをインストールしている場合のみです。

たとえば、ユーザーが別のデバイスで使っていたアプリを新しいデバイスにインストールした場合、最初のデバイスで使っていた設定や基本設定がすべて新しいデバイスに自動的に適用されます (ユーザーが同じ Microsoft アカウントを使って両方のデバイスにログインすることを前提としています)。それらの設定や基本設定に対する以降の変更もすべて自動的に移行されるため、デバイスに関係なく一貫したエクスペリエンスを得ることができます。セッション情報をローミング データとして保存すると、ユーザーは、あるデバイスで閉じたり中止したりしたアプリ セッションを、別のデバイスに切り替えた後に引き続き使うことができます。

注 **RoamingSettings**、**RoamingFolder**、**RoamingStorageQuota** の各プロパティは、Windows Phone 用には実装されていません。

注 このようなファイルは、**RoamingFolder** に配置されていてもローミングされません。

- フォルダーのように振る舞うファイルの種類。たとえば、拡張子が .zip や .cab のファイル
- 名前の先頭に空白のあるファイル
- 名前に次の Unicode 文字を含むファイル
  e794、e795、e796、e7c7、e816、e817、e818、e81e、e826、e82b、e82c、e831、e832、e83b、e843、e854、e855、e864、e7e2、e7e3、e7f3
- ファイル パス (ファイル名 + 拡張子) が 256 文字を超えるファイル
- 空のフォルダー
- 開いているハンドルがあるファイル

推奨と非推奨

- ユーザーの基本設定やカスタマイズ、リンク、小さなデータ ファイルにローミングを使います。たとえば、ローミングを使って、ユーザーの背景色の基本設定をすべてのデバイスで保持します。
- ユーザーがデバイス間で作業を続けられるようにローミングを使います。たとえば、下書きしたメールの内容やリーダー アプリで最近表示したページなどのアプリ データをローミングします。

- アプリ データを更新して、**DataChanged** イベントを処理します。このイベントは、クラウドからのアプリ データの同期が完了したときに発生します。
- 生データではなくコンテンツへの参照をローミングします。たとえば、オンライン記事のコンテンツではなく URL をローミングします。
- タイム クリティカルな重要な設定に対しては、**RoamingSettings** に関連付けられた HighPriority 設定を使います。詳しくは、アプリ データの概要をご覧ください。
- デバイス固有のアプリ データをローミングしないでください。ローカルにあるファイル リソースのパス名など、ローカルのみに関連した情報もあります。ローカル情報をローミングする場合は、その情報が別のデバイスで無効なときにアプリを回復できることを確認してください。
- 大量のアプリ データをローミングしないでください。アプリでローミングできるアプリ データの量には制限があります。この最大値を取得するには、**RoamingStorageQuota** プロパティを使ってください。 この制限に達した場合、アプリ データのサイズが制限を下回るまで、データはローミングできません。アプリを設計する際は、この制限を超えないようにサイズの大きいデータをどのように制限するかを検討してください。たとえば、ゲームの状態を保存するのにそれぞれ 10 KB 必要になる場合は、ユーザーによる保存を 10 ゲームまでに制限したりすると効果的です。
- 即時同期に依存するデータにローミングを使わないでください。Windows では即時同期が保証されません。ユーザーがオフラインであったり、待ち時間の長いネットワークを使っている場合、ローミングはかなり遅れる可能性があります。UI が即時同期に依存しないことを確認してください。
- 頻繁に変更されるデータにローミングを使わないでください。たとえば、再生中の曲の秒刻みの位置など、頻繁に変更される情報を追跡する場合は、この情報をローミング アプリ データとして保存しないでください。代わりに、現在再生中の曲など、変更の頻度が少なく、ユーザー エクスペリエンスも損なわないような情報を利用します。

その他の使い方のガイダンス

ローミングの前提条件

アプリ データのローミングは、Microsoft アカウントを使ってデバイスにログインするすべてのユーザーに利点をもたらします。ただし、いつでもデバイスでアプリ データのローミングを切り替えることができるのは、ユーザーとグループ ポリシーの管理者です。 ユーザーが Microsoft アカウントを使わない場合やデータのローミング機能を無効にする場合、ユーザーは引き続きアプリを使用できますが、アプリ データは各デバイスに対してローカルのままになります。

PasswordVault に格納されているデータは、ユーザーが "信頼" しているデバイスにしか移行されません。デバイスが信頼されていない場合、この資格情報コンテナーのセキュリティで確保されているデータはローミングされません。

競合の解決

アプリ データのローミングは、複数のデバイスでの同時使用を想定していません。 2 台のデバイスで特定のデータ単位が変更されたことが原因で同期中に競合が発生した場合、最後に書き込まれた値が常に優先されます。これにより、アプリで最新の情報が利用されます。データ単位が設定コンポジットの場合、競合の解決は設定の単位で行われ、最新の変更を含むコンポジットが同期されます。

データを書き込むタイミング

想定される設定の有効期間に応じて、データを書き込むタイミングを変える必要があります。変更の頻度が低いアプリ データや変更間隔の長いアプリ データは、変更されたらすぐに書き込むようにします。 ただし、頻繁に変更されるアプリ データは、アプリが中断されたとき以外は、一定の間隔 (5 分に 1 回など) でのみ書き込むようにします。 たとえば、音楽アプリでは、"現在の曲" の設定は新しい曲の再生が始まるたびに書き込みますが、曲の途中の実際の位置は中断したときにのみ書き込みます。

使いすぎに対する保護

リソースの不適切な使用を防止するために、システムにはさまざまな保護メカニズムが備わっています。アプリ データが想定どおりに移行されない場合は、デバイスが一時的に制限

されていることが考えられます。通常、この状況はしばらくすると自動的に解決されるため、操作は必要ありません。

バージョン

アプリ データは、バージョンに基づいてデータ構造をアップグレードできます。バージョン番号は、アプリのバージョンとは別の番号で、自由に設定することができます。強制ではありませんが、バージョン番号は新しいデータほど大きくすることを強くお勧めします。新しいデータを表すバージョン番号が小さくなると、データ損失などの望ましくない問題が発生する可能性があります。

アプリ データのローミングは、バージョン番号が同じインストールされたアプリの間でしか行われません。たとえば、どちらもバージョン 2 のデバイスの間やどちらもバージョン 3 のデバイスの間ではデータが移行されますが、バージョン 2 を実行中のデバイスとバージョン 3 を実行中のデバイスの間ではローミングは行われません。 他のデバイスでさまざまなバージョン番号を利用していたアプリを新たにインストールする場合、新たにインストールしたアプリは、最も大きいバージョン番号と関連付けられているアプリ データを同期します。

テストとツール

開発者は、ローミング アプリ データの同期をトリガーするためにデバイスをロックできます。 一定の期間にわたってアプリ データが移行されていない場合は、次の点を確認してください。

- ローミング データの最大サイズを超えていないこと (詳しくは、「**RoamingStorageQuota**」をご覧ください)。
- ファイルが閉じていて、適切に解放されていること。
- 同じバージョンのアプリを実行しているデバイスが 2 台以上あること。

開発者は Windows 8 Roaming Monitor を使用して、アプリのローミング状態を監視し変更することができます。

Windows ストア アプリと Windows Phone ストア アプリの間のデータのローミング

アプリの 2 つのバージョン (Windows ストア用のバージョンと Windows Phone 用のバージョン) を公開する場合、2 種類の異なるデバイスで実行されているアプリ間でアプリ データ

をローミングできます。異なる種類のデバイス上の異なるバージョンのアプリ間でデータをローミングするには、アプリの各バージョンに同じパッケージ ファミリ名 (PFN) を割り当ててください。

詳しくは、「Windows ストア アプリと Windows Phone ストア アプリ間でデータをローミングする方法」をご覧ください。

アプリから OneDrive へアクセスする場合のガイドライン

Microsoft OneDrive にあるユーザーのファイル、ドキュメント、画像、動画、フォルダー、アルバム、コメントを操作する Windows ストア アプリを設計する場合は、次のガイドラインに従ってください。

推奨と非推奨

OneDrive のユーザーは、Microsoft がデータのセキュリティとプライバシーを保護することを前提としています。また、重要なドキュメントの保管、写真の保存、友人とのエクスペリエンスの共有に OneDrive を利用します。アプリからデータにアクセスしやすければ、ユーザーにとっての OneDrive の価値が高まります。

ユーザーが OneDrive に対して持っている信頼感を維持するために、次の設計原則に従ってアプリを設計してください。

ユーザーがオプトインできるようにする

ユーザーは、アプリによるデータの処理方法を選べることや、アプリが自分のアカウントに接続する前に、アプリからアクセス許可を求められることを望みます。データが変更される場合は、事前に通知されることも望みます。こうした期待に応えるには、次のプラクティスに従ってください。

- 必ずユーザーによる明示的な要求または選択に基づいて OneDrive にファイルをアップロードする。

OneDrive に接続するアプリには、ユーザーが自分の意思で OneDrive にファイルをアップロードできるボタンを含める必要があります。アプリが既定で OneDrive にファイルを同期する場合は、データが保存される前にユーザーにこのことを知らせ、オプトインする機会を提供してください。

- アカウントへのユーザーのサインインとサインアウトには、アカウント チャームを使う。

  「Microsoft アカウントへのユーザーのサインインとサインアウト」で説明しているように、アプリでは、ユーザーが Microsoft アカウントにアクティブにサインインおよびサインアウトできる手段を提供する必要があります (ただし、ユーザーが既に Microsoft アカウントで Windows にサインインしている場合、アプリが明示的にユーザーをサインアウトさせることはできません)。

  詳しくは、「Microsoft アカウントのサインインの要件」をご覧ください。
- サインインしたユーザーが所有しているファイルにのみアクセスする。

  アプリで OneDrive ユーザー間でのファイルの共有を想定していない場合は、サインインしているユーザーのファイルにのみアクセスします。ユーザー自身が共有を選んだ場合にのみ、ユーザーと共有されているファイルやフォルダーにアクセスする必要があります。逆に、ユーザーの許可なく共有フォルダーにファイルを保存することはできません。
- ユーザーが OneDrive にデータを保存する場面について、選択肢を用意する。

  アプリでは、**Windows.Storage.Pickers** 名前空間を通じて Windows ファイル ピッカーを利用し、ファイルをユーザーの OneDrive に保存したり、OneDrive にあるファイルを開いたりできます。アプリが複数のファイルを同期する場合は、ユーザーのフォルダーに一意の名前のサブフォルダーを作成することを検討してください。ユーザーが OneDrive からファイルを開くときにファイル ピッカーを使う方法について詳しくは、「フォルダーとファイル」をご覧ください。

ユーザーのデータとプライバシーを保護する

アプリは OneDrive に対するユーザーの信頼を揺るがしてはなりません。ユーザーのデータは個別に扱ってください。ユーザーは、自分が選んだユーザーとのみファイルが共有されるものと考えます。重要な情報は、必要なときに使用できるように保管する必要があります。

重要 OneDrive オブジェクトに対してプログラムで設定したアクセス許可を、後からアプリが変更することはできません。

- ファイルを OneDrive にアップロードするときに、そのユーザーしかアクセスできないように既定で設定する。
  ユーザーがファイルの共有を明示的に要求した場合にのみ、ファイルを他のユーザーと共有します。
- 他のユーザーとのファイルへのリンクの共有に関して警告する。
  ユーザーがファイルへのリンクの共有を要求したら、アプリでその結果を知らせるようにしてください。特に、アプリでユーザーがファイルへの事前認証済みリンクを共有できるようにする場合は、そのリンクを受け取ったユーザーはだれでもファイルを確認できるようになることを知らせます。これらのリンクについてはファイルのアクセス許可は評価されず、リンクを開いたユーザーはだれでもコンテンツを表示できます。
  詳しくは、「OneDrive の中心となる概念」をご覧ください。
- リンクの用途に基づいて OneDrive オブジェクトへのリンクを作成する。
  可能な限り、埋め込みリンク、読み取り専用リンク、読み取り/書き込みリンクを共有するようにします。これらのリンクは、そのファイルを表示するアクセス許可があるユーザーだけが利用できます。ユーザーが特定のユーザーとフォルダーやファイルを共有する場合にのみ、ファイルへの事前認証済みリンクを提供します。これらのリンクについてはファイルのアクセス許可は評価されず、リンクを開いたユーザーはだれでもコンテンツを表示できます。
  詳しくは、「OneDrive の中心となる概念」をご覧ください。
- 既にあるファイルを上書きする場合、ユーザーに警告します。
  ファイルを OneDrive にアップロードしたときの既定の動作として、同じ名前を持つ既にあるファイルが上書きされます。競合がある場合は、既にあるファイルが上書きされることをユーザーに知らせます。**Overwrite** ヘッダーを追加して "false" に設定すると、既にあるファイルは上書きされません。

意図したとおりに OneDrive と Windows を使う

OneDrive を通じて自由に使える記憶域を、あらゆるデータを保管するクラウド データ ソリューションとして利用することは魅力的です。OneDrive には Windows ストア アプリ向け

のさまざまなオプションがありますが、意図したとおりに使われる場合に、アプリに対するメリットが最も大きくなります。OneDrive は、任意のデバイスからドキュメント、写真、その他重要情報にアクセスできるように設計されています。

- ドキュメントの保存、表示、編集や、フォト アルバムの作成と共有に OneDrive を使う。

  OneDrive は、スケーラブル データベースの格納、構成ファイルの共有、Web アプリケーションのホストなどを行わずに済む手段として利用されます。 ユーザーの個別のファイルを簡単に格納、共有する目的でのみ使われます。
- ファイルのアップロードの前に OneDrive にスペースがあることを確認する。

  OneDrive のユーザーごとに、使用可能な記憶域の量に制限があります。ユーザーのアカウントに対する割当量を超える場合は、アプリでファイルを保存しようとすると、呼び出しエラーが返されます。OneDrive にファイルを保存する前にユーザーの利用可能な記憶域をチェックすることをお勧めします。

  OneDrive の利用可能な領域をチェックする方法について詳しくは、「一般的なタスク」をご覧ください。
- 組み込みの Windows 機能を使う。

  可能な限り、OneDrive のホストまたは操作には Windows 機能と Windows UI を使います。たとえば、ファイルを開く処理や保存には **Windows.Storage.Pickers** 名前空間で提供されるファイル ピッカーを使います。また、ユーザーのさまざまなデバイスで少量のデータを保存する場合は、Windows アプリケーション データ API を使います。

  OneDrive からファイルを開くときにファイル ピッカーを使う方法について詳しくは、「フォルダーとファイル」をご覧ください。

  Windows アプリケーション データ API を使う方法について詳しくは、「アプリケーション データのローミングのガイドライン」と「クイック スタート: アプリケーション データのローミング」をご覧ください。

その他の使い方のガイダンス

OneDrive は、ユーザーがファイルをクラウドに保存し、アクセスできる信頼性の高い場所です。ユーザーは、Microsoft アカウントを使ってサインインすると、好きな Windows デ

バイスから OneDrive にある自分のファイルにアクセスできます。OneDrive では 7 GB の記憶域が無料で提供され、写真、ドキュメント、動画、オーディオ ファイルの保存と共有に利用できます。

Windows ストア アプリは、OneDrive にあるファイルやフォルダーへのアクセスをユーザーに提供することができます。OneDrive への接続を利用すると、ハード ディスクを散らかさずに、OneDrive にあるファイルを操作 (開く、読み取り、保存、ダウンロード) できるアプリを開発できます。OneDrive API は、Windows ストア アプリからの使用を想定して設計され、アプリの設計にスムーズに統合されます。

OneDrive を使うための設計

広い意味で、OneDrive は、個々のファイルとやり取りするすべてのアプリの役割を果たすことができます。アプリにファイルの操作機能 (読み取り、表示、保存、ダウンロード、開く) がある場合は、アプリの設計に OneDrive を追加できます。OneDrive は、追加のコードを大量に記述しなくても、Windows の組み込みの機能を利用して、Windows ストア アプリのアーキテクチャとうまく統合できます。

重要 OneDrive API は Live Connect SDK に含まれています。OneDrive に接続する Windows ストア アプリの開発を始める前に、Live Connect SDK をインストールし、プロジェクトに SDK への参照を追加する必要があります。

- Live Connect SDK をダウンロードするには、Live Connect SDK のダウンロード ページにアクセスしてください。
- OneDrive API のドキュメントを確認するには、「OneDrive for Developers (開発者向け OneDrive)」と「OneDrive API」をご覧ください。

Microsoft アカウントへのユーザーのサインインとサインアウト

もちろん、OneDrive とやり取りするどのアプリでも、OneDrive に関連付けられた Microsoft アカウントを使ってユーザーがサインインとサインアウトを実行できるようにする必要があります。アプリ自体の設計ではないものの、アカウントへのユーザーのサインインは、OneDrive と統合されるアプリを作成するうえで重要な手順です。

ユーザーのサインインに関しては、アプリの設定チャームにアカウントのページとプライバシーに関する声明のページを作成することをお勧めします。アカウント ページには、ユーザーがアカウントへのサインインとサインアウトを行うためのボタンを設けます。アプリの残りのサインイン プロセスは Windows UI で処理します。

詳しくは、次のリソースをご覧ください。

- Microsoft アカウントへのユーザーのサインインについて詳しくは、「ユーザーのサインイン」をご覧ください。
- JavaScript を使った Windows ストア アプリ向けに設定チャームをカスタマイズする方法について詳しくは、「Microsoft アカウントのサインインの要件」をご覧ください。
- C# または Visual Basic と XAML を使った Windows ストア アプリ向けに設定チャームをカスタマイズする方法について詳しくは、「Microsoft アカウントのサインイン要件」をご覧ください。

OneDrive への新しいファイルの保存と既にあるファイルの更新

一部のユーザーにとっては、OneDrive が "マイ ドキュメント" です。ファイルの保存に OneDrive を使うユーザーのために、アプリでは OneDrive にユーザーのデータを保存するオプションを提供できます。たとえば、ユーザーがアプリで新しいファイルを作成したときに、保存場所として OneDrive を提供できます。 ユーザーがアプリでファイルを編集した場合は、その編集内容も OneDrive に保存できます。

実際のところ、ユーザーが新しいファイルを作ることができるアプリでは、OneDrive へのユーザー アクセスを提供することには大きな意味があります。

- OneDrive と統合されるアプリを作成する方法のガイドラインについては、「推奨と非推奨」をご覧ください。
- ユーザーの OneDrive から画像、動画、オーディオ ファイルをアップロードする方法について詳しくは、「アルバム、写真、動画、オーディオ、タグ」をご覧ください。
- ユーザーの OneDrive でファイルを保存および更新する方法について詳しくは、「フォルダーとファイル」をご覧ください。

OneDrive からのファイルのダウンロード、オープン、表示

既に述べたように、一部のユーザーはクラウドに多くのデータを保存します。ユーザーはクラウドのデータを表示できることを望みます。アプリでは、OneDrive からファイルを開いて読むオプションを提供できます。アプリは、ユーザーが確認するファイルのコンテンツをダウンロードして開き、表示できます。

たとえば、動画を再生するアプリの場合、ユーザーが OneDrive のフォルダーから映画を開く機能を提供できます。ユーザーが特定の種類のファイルを開いて表示できるリーダー アプリもあります。

注 Windows ストア アプリでは、OneDrive のファイルを表示するだけにとどまらない機能を提供することをお勧めします。Windows には最初から OneDrive アプリがインストールされています。アプリに独特の機能があれば、ユーザーにダウンロードしてインストールしてもらえる可能性が高まります。

詳しくは、次のリソースをご覧ください。

- OneDrive と統合されるアプリを作成する方法のガイドラインについては、「ベスト プラクティス」をご覧ください。
- ユーザーの OneDrive から画像、ビデオ、オーディオ ファイルをアップロードする方法について詳しくは、「アルバム、写真、動画、オーディオ、タグ」をご覧ください。
- ユーザーの OneDrive でファイルを保存および更新する方法について詳しくは、「フォルダーとファイル」をご覧ください。

シングル サインオンと接続されているアカウントのガイドライン

ユーザー認証シナリオ

ユーザーは、Microsoft アカウントの資格情報を使って、Windows 8 を実行しているデバイスにサインインできます。このとき Windows 8 では、Windows ストア アプリを使ってユーザーに対して認証されたエクスペリエンスを有効にできます。これらのエクスペリエンスには次のものがあります。

- ユーザーは最もよく使うオペレーティング システム設定を Microsoft アカウントと関連付けることができます。これらの設定は、Windows 8 を実行していてクラウドに接続しているデバイスに、ユーザーがそのアカウントを使ってサインインするときは常に使用できます。ユーザーがサインインすると、デバイスはユーザー設定をクラウドから自動的に取得することを試みてから、そのユーザー設定をデバイスに適用します。
- Windows ストア アプリではユーザー固有の設定を格納できるため、Windows 8 を実行する任意のデバイスで設定をローミングできます。オペレーティング システム設定と同じように、このようなユーザー固有のアプリ設定は、クラウドに接続している Windows 8 を実行中のデバイスに、ユーザーが同じ Microsoft アカウントを使ってサインインするときは常に使用できます。ユーザーがサインインすると、アプリがインストールされている場合、そのデバイスは設定をクラウドから自動的に取得し、デバイスに適用します。
- Windows 8 では、ユーザーは Microsoft アカウントを任意のアプリまたは Web サイトのサインイン資格情報と関連付けることができるため、Windows 8 が実行されるすべてのデバイスでその資格情報をローミングできます。ユーザーがそのアカウントを使って Windows 8 が実行中のデバイスにサインインして、アプリを実行するか Web サイトを訪問した場合、そのユーザーに対応するサインイン資格情報が格納されていれば、Windows 8 によって自動的にそのユーザーのサインインが試行されます。
- ユーザーが Microsoft アカウントを使って Windows 8 が実行中のデバイスにサインインするとき、やはり認証に Microsoft アカウントを使うデバイスで実行されているアプリやサービスは、そのユーザーの Microsoft アカウントを使ってサインインし、そのユーザーが共有することを同意したデータを取得することができます。

Microsoft アカウント認証 API を使う状況

次の質問に "はい" と答えれば答えるほど、アプリ (およびアプリのコンパニオン Web サイト) と Microsoft アカウント認証 API の統合を検討する必要があります。

- 使っているアプリが Windows ストア アプリか
- アプリはユーザーに合わせたエクスペリエンスを提供するか
- アプリは、独自のクラウド サービスや、Outlook.com、Microsoft OneDrive、Windows Live Messenger のような Microsoft クラウド サービスにアクセスするか
- アプリでユーザー認証システムを提供する必要があるものの、自分で作るための時間、知識、インフラストラクチャがないか

Microsoft アカウント認証 API のメリットを受けられる、アプリとコンパニオン Web サイトのカテゴリがいくつかあります。

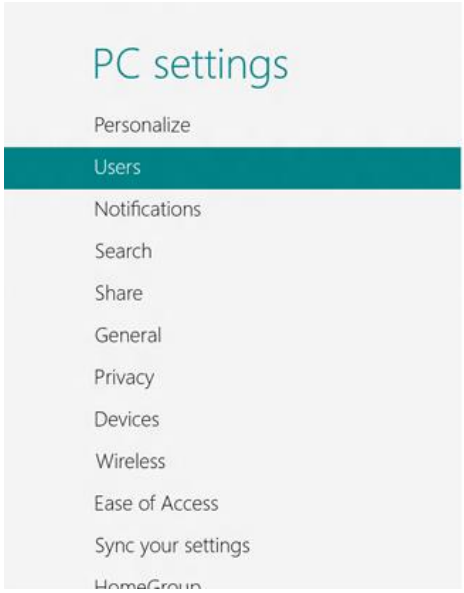
- **独自のクラウド サービスにアクセスし、ユーザー認証が必要なアプリ。**アプリがクラウド サービスにアクセスし、ユーザーを認証する必要がある場合、コードはアプリまたは Web サイトがユーザーの代わりにクラウド サービスにアクセスできるようにする認証トークンを要求できます。Microsoft アカウント認証 API は、使用するクラウド サービスに対応する JavaScript Object Notation (JSON) 形式の認証トークンを生成します。それぞれの認証トークンには、アプリ固有のユーザー ID が含まれています。Microsoft アカウント認証 API を使って特定のクラウド サービスで使用する JSON 形式の認証トークンを生成するには、「**OnlineIdServiceTicketRequest**」をご覧ください。
- **Outlook.com、OneDrive、Messenger などの Microsoft クラウド サービスにアクセスするアプリとコンパニオン Web サイト。**アプリまたはアプリのコンパニオン Web サイトが Outlook.com や OneDrive のユーザー データにアクセスする場合や、ユーザーが Messenger 仲間とやり取りできるようにする場合は、Live Connect API によって認証トークンの複雑さが整理され、クラウド サービスを操作するコードをある程度簡単に記述できるようになります。

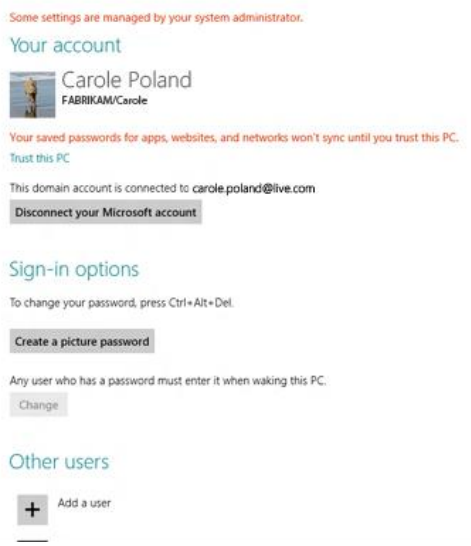
Windows ストア アプリへのユーザー認証機能の追加

Windows ストア アプリ用に一貫したユーザー サインインおよびサインアウトのコードを作るには、「Microsoft アカウントのサインインの要件」と「ログインのガイドライン」をご覧ください。

注意 Windows 8 デバイスでは、ユーザーは Microsoft アカウントに関連付けられた (または接続された) ローカルまたはドメインの Windows アカウントを使ってそのデバイスにサインインできます。その後、ユーザーはサインイン用のこの特定の Microsoft アカウントに依存する Windows ストア アプリを実行できます。その場合、前に示したトピックで説明したガイドラインを使って後でアプリからユーザーをサインアウトしようとしても、ユーザーはそのアプリから正常にサインアウトされません。これを防ぐには、アプリでユーザーがサインアウトされないようにするために、サインアウト コマンドを非表示にする必要があります。代わりに、ユーザーは次の 2 つの方法でアプリからサインアウトすることができます。

- ユーザーは Microsoft アカウントのローカルまたはドメインの Windows アカウントとの関連付けを解除 (または切断) することができます。これを行うには、**[PC 設定]** (次のスクリーン ショットをご覧ください) で **[ユーザー]**、**[Microsoft アカウントの関連付けを解除する]**、**[完了]** の順にタップします。





- ユーザーは他のアカウントを使うように切り替えることができます。これを行うには、**[スタート]** 画面で、アカウントの画像、**[アカウントの切り替え]** の順にタップした後、他のアカウントの資格情報を使ってサインインします。

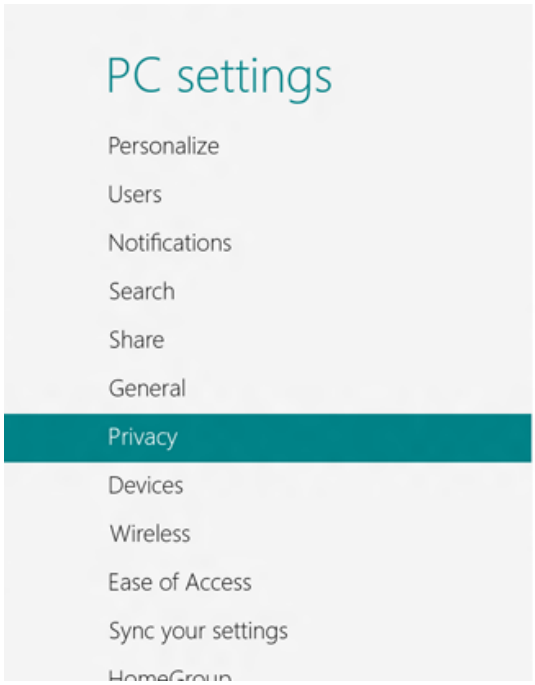
ユーザーがこのどちらかの操作を実行すると、サインイン用の特定の Microsoft アカウントに依存するすべての Windows ストア アプリからも自動的にサインアウトされます。

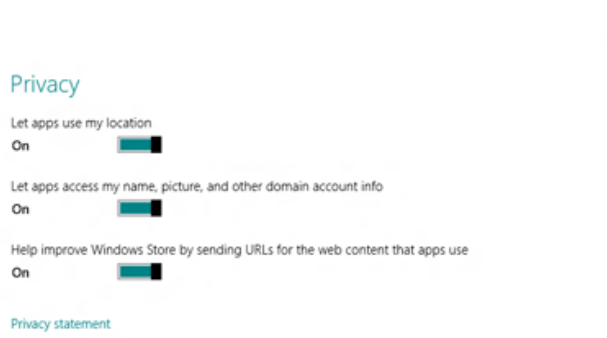
ユーザーに制御を委ねる

Windows 8 を実行中のデバイスに、Microsoft アカウント (または Microsoft アカウントに接続されているローカルまたはドメインのユーザーの Windows アカウント) を使ってサインインすると、ユーザーは Microsoft アカウント認証 API を使うアプリまたはコンパニオン Web サイトがユーザーのために操作できる範囲を制御します。たとえば、ユーザーが Outlook.com や OneDrive のような Microsoft クラウド サービスにアクセスする参加アプリにサインインすると、アプリまたはアプリのコンパニオン Web サイトが特定のクラウド サービスからユーザーの関連データまたはファイルにアクセスすることを許可する (同意する) ように求めるメッセージが表示されます。

しかし、アプリが Outlook.com や OneDrive などの Microsoft クラウド サービスのデータにアクセスするのではなく、独自のクラウド サービスにアクセスする場合は、ユーザーが **[PC 設定]** の **[プライバシー]** でこの設定を明示的に変更していない限り、同意を求めるメッセージは表示されません。それでも、参加アプリおよび Web サイトは、Microsoft アカウント認証 API を使ってアプリまたは Web サイト独自のユーザー ID を取得できます。そのため、アプリまたは Web サイトはそのユーザー ID にデータを関連付けることができます。次にユーザーがサインインすると、アプリまたは Web サイトは同じユーザー ID を取得し、そのユーザー ID を使って既にユーザーに関連付けられているデータをすべて取得できます。

Windows 8 では、ユーザーのアカウントに関連付けられたアプリの名前、画像などのデータに対するアクセスを、ユーザーがすべてのアプリで制限できます。これを行うには、**[PC 設定]** (次のスクリーン ショットをご覧ください) で、**[プライバシー]** をタップし、**[アプリで自分の名前、画像、その他のドメイン アカウント情報にアクセスすることを許可する]** を **[オフ]** にスライドします (このオプションは既定ではオンになっています)。





ユーザーのサインインとサインアウト ステータスの変化への応答

Microsoft アカウント認証 API を使うアプリは、ユーザーがその Microsoft アカウントをローカルまたはドメインの Windows アカウントに接続または切断するたびに適切に応答する必要があります。ユーザーがこれを行うと、**ConnectedStateChange** バックグラウンド タスクが開始されます。 このタスクが開始されるたびに、参加アプリはユーザーのアプリ用の ID がクリアされたかどうかをチェックする必要があります。

- ユーザーのアプリ用の ID がクリアされている場合、アプリではまず、ユーザーの関連するすべてのタイル通知をクリアする必要があります。 アプリでユーザーがサインイン中であるかどうかを表示している場合、アプリの状態を変更して、ID がクリアされたユーザーがもうサインインしていないことを示す必要があります。ただし、アプリを既定の状態にリセットしないでください。代わりに、ユーザーがサインアウト状態でなければ、ユーザーはまだアプリを使っていて、作業を中断した時点から再開するつもりであると想定する必要があります。 唯一の違いは、サインアウトしたユーザーはタイル通知などのクラウド サービスにアクセスできなくなる点です。 この方法を使った場合の動作はアプリによって異なることに注意してください。使い続けるためにユーザーに再サインインを求める必要があるアプリもありますが、サインアウト状態でも動作を継続できるアプリもあります。

注 サインアウトしたユーザーのためにアプリ固有のデータをクラウドに格納しないアプリの場合、動作はアプリによって異なります。そのユーザーのアプリ固有のデータが表示されないように、すべてクリアすることをお勧めします。ただし、ユーザーが Microsoft アカウントとローカルまたはドメインの Windows アカウントの関連付けを再接続する場合に備えて、アプリでこのデータをローカルに保存する必要があります。

- ユーザーのアプリ用の ID がクリアされていない場合、ユーザーはアプリで既に使っているのと同じアカウントに接続されているため、これ以上アプリで実行する必要があることは何もありません。

ConnectedStateChange バックグラウンド タスクについて詳しくは、**OnlineIdConnectedStateChange** システム イベントの **SystemTriggerType** 列挙に関するページをご覧ください。

ユーザーのアプリ用の ID がクリアされたかどうかのチェック方法について詳しくは、**authenticatedSafeCustomerId** プロパティに関するページをご覧ください。

サムネイルのガイドライン

このガイドラインでは、サムネイル イメージを使って、Windows ストア アプリでファイルを参照するユーザーに最適なプレビューを提供する方法について説明します。

アプリにサムネイルを含めるかどうか

アプリでユーザーがファイルを参照できるようにする場合は、サムネイル イメージを表示して、ファイルをすばやくプレビューできるようにします。

サムネイルは次の場合に使います。

- 多くの項目 (ファイルやフォルダーなど) に対するプレビューを表示する場合。たとえば、フォト ギャラリー アプリでは、ユーザーが写真ファイルを参照するときにサムネイルを使って各写真が小さく表示されるようにします。
- 個別の項目 (個々のファイルなど) に対するプレビューを表示する場合。たとえば、ユーザーがファイルを開くかどうかを決める前に、より見やすい大きなサムネイルと共に、ファイルの詳しい情報を表示できます。

推奨と非推奨

- サムネイルを取得するメソッドを呼び出すときに、サムネイル モード (**picturesView**、**videosView**、**documentsView**、**musicView**、**listView**、**singleItem**) を指定します。これにより、ユーザーが参照するファイルの種類を表示するのに最適なサムネイル イメージが用意されます。
- ファイルの種類に関係なく単一項目用のサムネイルを取得するには、**singleItem** モードを使います。その他のサムネイル モード (**picturesView**、**videosView**、**documentsView**、**musicView**、**listView**) の目的は、複数ファイルのプレビューを表示することです。
- サムネイルの読み込み中は、サムネイルの代わりに汎用のプレースホルダー イメージを表示します。このようにプレースホルダーを使うことで、ユーザーはプレビ

ユー イメージを読み込む前に項目を操作できるため、アプリの見かけの応答速度を高めることができます。

プレースホルダー イメージは次の条件を満たす必要があります。

- 代わりとなる項目の種類に固有である。たとえば、フォルダー、画像、動画にはすべて、それぞれ異なるアイコン、テキスト、色を使った専用のプレースホルダーを用意する必要があります。
- 代わりとなるサムネイル イメージとサイズおよび縦横比が同じである。
- サムネイル イメージが読み込まれるまで表示される。サムネイルを取得できない場合は、代わりにプレースホルダー イメージを表示します。

- テキスト ラベル付きのプレースホルダー イメージを使って、フォルダーとファイル グループを表現します。こうすることで、フォルダーやファイル グループのようなシステム構造と個別のファイルを区別することができます。これらの項目の種類をそれぞれ視覚的に区別することで、アプリでユーザーが参照しやすくなります。フォルダーの名前、またはファイル グループを作るために使われた条件を、テキスト ラベルとしてプレースホルダーに含めます。
- 項目 (ファイル、フォルダー、ファイル グループなど) のサムネイルを取得できない場合はプレースホルダー イメージを表示します。
- ドキュメントと音楽ファイルのプレビューを表示するときは、サムネイル イメージに加えてファイル情報も表示します。これによってユーザーは、サムネイル イメージだけからはすぐに得られない可能性のある、ファイルに関する重要な情報を識別できます。たとえば、アルバム アートを示すサムネイルと並べて音楽ファイルのアーティスト名を表示したりできます。
- 画像ファイルと動画ファイルのサムネイルと共に追加のファイル情報を表示しないでください。ほとんどの場合、ユーザーが画像と動画を参照するには、サムネイル イメージだけで十分です。

その他の使い方のガイダンス

推奨サムネイル モードとその特徴

プレビューの表示対象	サムネイル モード	取得するサムネイル イメージの特徴
画像 動画	picturesView videosView	**サイズ:** 中、190 以上を推奨 (画像サイズが 190 × 130 の場合) **縦横比:** 均一な横長の縦横比 (約 0.7) (サイズが 190 の場合は 190 × 130) プレビューの場合はトリミング 縦横比が統一されているため、画像をグリッド内で揃えるときに便利です。
ドキュメント 音楽	documentsView musicView listView	**サイズ:** 小、40 × 40 ピクセル以上を推奨 **縦横比:** 均一な正方形の縦横比 縦横比が正方形であるため、アルバム アートのプレビューに最適 ドキュメントは、ファイル ピッカーのウィンドウと同じように表示 (同じアイコンを使用)
任意の 1 つの項目	singleItem	**サイズ:** 大、長辺が 256 ピクセル以上 **縦横比:** 可変、ファイルの元の縦横比を使用

ヒント 今後、各モードのサムネイル イメージの特徴は、さらに具体的になる可能性があります。それに対応するために、プレビューを表示するファイルの種類に最も近いサムネイル モードを指定することをお勧めします。たとえば、動画ファイルを表示する場合は、**videosView** サムネイル モードを使います (ただし単一の動画ファイルを表示する場合は、singleItem モードを使います)。。

推奨サムネイル モードを使う理由

以下の例は、取得したサムネイル イメージが、ファイルの種類とサムネイル モードに応じてどのように異なるかを示します。

	指定するサムネイル モード		
項目の種類	取得時に使ったモード： • picturesView • videosView	取得時に使ったモード： • documentsView • musicView • listView	取得時に使ったモード： • singleItem
画像		サムネイルは縦横比が正方形になるようにトリミングされています。 	サムネイル イメージには、ファイルの元の縦横比が使われます。 
動画	サムネイル イメージには、画像と区別するための装飾が追加されます。 	サムネイルは縦横比が正方形になるようにトリミングされています。 	サムネイル イメージには、ファイルの元の縦横比が使われます。 

ミュージック	サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります (関連付けられたアプリが Windows ストア アプリの場合は、アプリのタイルの背景色が使われます。)。 	ファイルにアルバム アートが含まれる場合、サムネイルはアルバム アートになります。 それ以外の場合、サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります (関連付けられたアプリが Windows ストア アプリの場合は、アプリのタイルの背景色が使われます。)。 	ファイルにアルバム アートが含まれる場合、サムネイルはアルバム アートになり、ファイルの元の縦横比が使われます。それ以外の場合、サムネイルはアイコンです。 
ドキュメント	サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります。(関連付けられたアプリが Windows ストア アプリの場合は、アプリのタイルの背景色が使われます。) 	サムネイルは、適切なサイズの背景に配置されたアイコンです。背景色は、ファイルに関連付けられたアプリによって決まります (関連付けられたアプリが Windows ストア アプリの場合は、アプリのタイルの背景色が使われます。)。 	ドキュメントのサムネイルがある場合は、そのサムネイルが表示されます。  それ以外の場合、サムネイルはアイコンです。 

フォルダー	フォルダーに画像ファイルが含まれる場合は、画像のサムネイルが使われます。  それ以外の場合、サムネイル イメージは取得されません。	サムネイル イメージは取得されません。	サムネイルは、フォルダーを表すアイコンです。
ファイル グループ	グループ内のファイルに画像ファイルが含まれる場合は、画像のサムネイルが使われます。  それ以外の場合、サムネイル イメージは取得されません。	グループ内のファイルにアルバム アートを含むファイルがある場合、サムネイルはアルバム アートになります。  ファイル グループにアルバム アートが存在しない場合、サムネイル イメージは取得されません。	グループ内のファイルにアルバム アートを含むファイルがある場合、サムネイルはアルバム アートになり、ファイルの元の縦横比が使われます。  それ以外の場合、サムネイルはファイルのグループを表すアイコンです。 

ユーザー名とアカウントの画像のガイドライン

説明

Windows 8 の場合、アプリで現在のユーザーの名前と画像 (アカウントの画像) を取得し、それらを使って、ユーザーを識別し、ユーザーに合わせたエクスペリエンスを作成できます。たとえば、メッセージング アプリで、名前とアカウントの画像を使って、ユーザーを会話の参加者として識別することや、ゲーム アプリで、それらを使って、ゲームのスコアボード ページでユーザーをプレイヤーとして識別することなどができます。

推奨と非推奨

- アプリケーションのインターフェイスに適した画像サイズを使います。
- 状態は、ステータス バーを使ってアカウントの画像のごく一部として表示します。
- アカウントの状態のアイコンには、背後に表示されるアカウントの画像に紛れないようなアイコンを使ってください。

その他の使い方のガイダンス

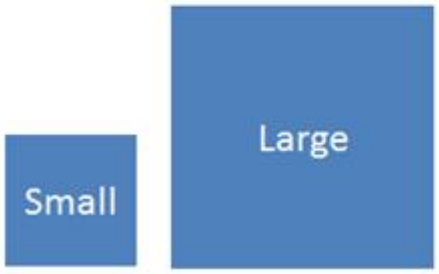
画像を撮影できるアプリの場合は、アプリをアカウントの画像プロバイダーとして宣言することを検討してください。これを行うと、アプリが **[PC 設定]** の **[パーソナル設定]** にある **[アカウントの画像]** ページに表示されます。ここからアプリを選んで、新しいアカウントの画像を作成できます。

詳しくは、「アプリ コントラクトと拡張機能」と「**UserInformation**」をご覧ください。アカウントの画像名のサンプルに関するページもご覧ください。

アカウントの画像のサイズ

アプリでは、アカウントの画像を小さな画像、大きな画像、またはビデオ (動的な画像) として取得できます。画像は、最大 DPI プラトーで適切に表示されるサイズに調整されます (1.8 倍)。

Small	Large
96x96	448x448



Plateau	Small	Large
1.0	48	224
1.4	67	314
1.8	86	403

アカウントの画像のビデオは、フレーム サイズが 448x448、最長時間が 5 秒、最大サイズが 5 MB です。

アカウントの画像が特定のエクスペリエンスの中心部分ではない場合、たとえば多数のユーザーを一覧表示するときには、小さな画像を使います。アプリ領域内でごく一部のユーザーを識別する必要がある場合や、各ユーザーを明確に識別する必要がある場合は、大きな画像を使います。たとえば、メッセージング アプリの会話の参加者や、電話の着信の呼び出し元の画像を表示する場合に、大きな画像を使います。

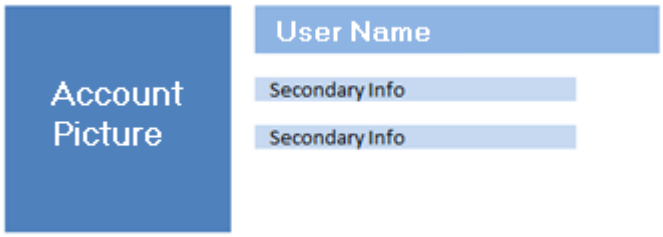
ユーザー名

アプリでは、**UserInformation** クラスのメンバーを使って、ユーザー名を取得できます。ユーザーの姓と名は、ユーザーの表示名と同じように、別々に使うことができます。表示名は、ロケールに適した順番になるように自動的に書式設定されます。

アカウントの画像とユーザー名の同時表示

アカウントの画像とそれに対応するユーザー名を一緒に表示する際は、次のいずれかの形式を使うことをお勧めします。

- **左右に並べる**—十分なスペースがある場合は、この形式がユーザーにとって最も読みやすいため、アカウントの画像とユーザー名を左右に並べて表示します。 次の例は、お勧めの要素の配置を示しています。ユーザー名がアカウントの画像の上端に揃えて配置され、補助的な情報がユーザー名よりも小さいフォント サイズで表示されています。



- **上下に並べる**—スペースが限られている場合は、アカウントの画像の下にユーザー名を表示します。ただし、ユーザー名全体が 1 行に収まらない場合があるため、注意してください。ユーザー名がアカウントの画像の左端に揃えて配置され、補助的な情報がユーザー名よりも小さいフォント サイズで表示されています。

状態の表示

アプリでアカウントの画像と状態情報を組み合わせる必要がある場合は、ステータス バーまたはアイコン オーバーレイ を使うことをお勧めします。ステータス バーの方が画像に被らず、目立つため、お勧めです。ステータス バーには、アカウントの画像と簡単に区別できるだけの幅が必要です。 ステータス バーの例を次に示します。

アイコン オーバーレイを使うと、もっと複雑な状態を伝えることができます。アイコン オーバーレイは、画像の細部に埋もれる可能性があるため、さまざまな背景ではっきり目立つアイコンを使うようにしてください。アイコン オーバーレイの例を次に示します。

起動、中断、再開のガイドライン

このセクションでは、魅力的な起動エクスペリエンスの作成と、ユーザーが切り替えたときに一時停止し、元に戻すと再開するアプリの設計に関するガイドラインを示します。

このセクションの内容

トピック	説明
アプリの起動と中断	Windows ランタイム アプリの一時停止と再開の動作を設計するときは、次のガイドラインに従ってください。
スプラッシュ スクリーン	ユーザーに快適な起動エクスペリエンスを提供するために、スプラッシュ スクリーンをカスタマイズし、スプラッシュ スクリーンを拡張するには、以下のガイドラインに従ってください。

アプリの中断と再開のガイドライン

ユーザーが切り替えたときに一時停止し、元に戻すと再開するアプリを設計します。アプリの目的と使用パターンを慎重に検討して、アプリが一時停止および再開されるときに可能な限り最適な操作性を実現できるようにします。 Windows ランタイム アプリの一時停止と再開の動作を設計するときは、次のガイドラインに従ってください。

Windows ランタイム アプリのライフサイクルの概要については、「アプリのライフサイクル」を参照してください。

注 Windows 8.1 と Windows Phone 8.1 でシステムの応答性を向上させるために、アプリには中断後にリソースへの優先度の低いアクセスが与えられます。この新しい優先度をサポートするために、中断操作のタイムアウトが延長され、アプリには通常の優先度と同程度のタイムアウト (Windows では 5 秒、Windows Phone では 1 ～ 10 秒) が与えられます。 このタイムアウトの時間枠を延長したり、変更したりすることはできません。

推奨と非推奨

- 短時間の後に再開した場合は、中断したときのアプリの状態に戻します。たとえば、ユーザーが電子メールを書いている途中で別のアプリに移動し、電子メール

アプリに戻った場合、メール アプリのメイン ランディング ページではなく、書きかけのメールのページに戻る必要があります。

- 長時間の後に再開した場合、ユーザーはアプリの既定のランディング ページに戻ります。たとえば、時間がたった記事や以前の天候データの表示に戻るのではなく、ニュースや天気予報アプリの主なランディング ページに戻ります。
- 適切な場合は、アプリの以前の状態を復元するか、アプリを新たに開始するかをユーザーが選択できるようにします。たとえば、ゲームに戻るときに、ゲーム アプリを再開するか、新たに開始するかを選べるようにプロンプトを表示できます。
- アプリを一時停止するときにアプリ データを保存します。一時停止中のアプリは、システムによって終了されても通知を受け取りません。したがって、アプリの状態を復元できるように、データを明示的に保存することが重要です。
  アプリでセカンダリ タイルなどの複数の起動ポイント、トースト通知、ファイルと URI の関連付けをサポートしている場合は、それぞれ起動ポイントに別々のナビゲーション履歴を作成することを検討してください。中断時には、プライマリ起動ポイントに関連付けられた状態を保存し、状態が失われることにユーザーが不満を感じるシナリオでのみセカンダリ起動ポイントの状態を保存します。保存する状態が多すぎると、アプリの再開が遅くなる可能性があります。
- 保存されたアプリ データを使ってアプリを復元します。
- アプリが中断されているときに、排他リソースとファイル ハンドルを解放します。前に説明したように、一時停止中のアプリは終了時に通知を受け取らないため、アプリを一時停止する際は他のアプリからアクセスできるように、リソースとハンドル (Web カメラ、I/O デバイス、外部デバイス、ネットワーク リソースなど) を確実に解放します。
- ユーザーが最後に表示した後にコンテンツが変更された場合は、UI を更新します。再開されたアプリは、ユーザーが切り替えている間も実行されていたように表示されます。
- アプリが画面から消されても終了しないでください。オペレーティング システムでは、ユーザーは一貫した方法でアプリにアクセスして管理できます。アプリは画面から消されると一時停止します。アプリのライフサイクルをシステムにゆだねて、ユーザーができるだけ効率的にアプリに戻れるようにします。これにより、システムのパフォーマンスとデバイスのバッテリ寿命も最適な状態になります。

- ユーザーによって明示的に閉じられたアプリの状態は復元しないでください。回復不能の状態におちいったために、ユーザーがアプリを閉じた可能性があります。アプリがユーザーによって明示的に閉じられた場合は、再開のエクスペリエンスではなく新しいエクスペリエンスを提供します。アプリがユーザーによって閉じられた場合、**PreviousExecutionState** プロパティの値が **ClosedByUser** になります。
- クラッシュが発生したために終了したアプリの状態は復元しないでください。アプリが予期せずに終了した場合、保存されているアプリ データは壊れている可能性があります。この格納されたデータを使ってアプリを以前の状態に復元しないようにします。
- [閉じる] ボタンを表示するなど、ユーザーが UI からアプリを終了できる方法を提供しないようにします。ユーザーは、システムがアプリを管理しているという安心感を持っています。システムはアプリを自動的に終了することで、最良のシステム パフォーマンスと信頼性を確保できます。またユーザーは Windows でジェスチャを使うか、Windows Phone でタスク スイッチャーを使って、アプリを閉じることができます。
- ディープリンクされたページに、ユーザーが取り残されることがないようにします。ユーザーがプライマリ タイル以外の起動ポイントからアプリを起動し、ディープ リンクされたページにランディングした場合でも、ユーザーがアプリのトップ ページに移動できるように UI を用意します。または、プライマリ タイルをタップしてトップ ページにアクセスできるようにします。

スプラッシュ スクリーンのガイドライン

ユーザーに快適な起動エクスペリエンスを提供するために、スプラッシュ スクリーンをカスタマイズし、スプラッシュ スクリーンを拡張するには、以下のガイドラインに従ってください。

推奨と非推奨

- スプラッシュ スクリーンをカスタマイズして、アプリを特徴付けます。
  スプラッシュ スクリーンは、画像と背景色で構成され、どちらもカスタマイズできます。スプラッシュ スクリーンが適切にデザインされていると、アプリがより魅力的なものになります。
  画像と背景色を組み合わせてスプラッシュ スクリーンを作ると、アプリがインストールされているデバイスのフォーム ファクターに関係なく、スプラッシュ スクリーンが適切に表示されます。スプラッシュ スクリーンが表示されるとき、背景のサイズだけがさまざまな画面サイズに合わせて変更されます。画像のサイズは常に変わりません。
  このスプラッシュ スクリーンを拡張してカスタマイズする方法については、「クイック スタート: スプラッシュ画面の追加」をご覧ください。
- アプリのランディング ページを表示する前に追加のタスクを完了できるように、スプラッシュ スクリーンを拡張します。
  Windows で表示されるスプラッシュ スクリーンに似たアプリのスプラッシュ スクリーン ページを作成することで、アプリの読み込みエクスペリエンスをさらに制御できます。システムで表示されるスプラッシュ スクリーンに似せることで、スムーズで内容の伝わる読み込みエクスペリエンスをユーザーに提供できます。アプリの UI の準備やネットワーク データの読み込みに時間がかかる場合、アプリがこれらのタスクを完了している間、スプラッシュ スクリーンを拡張することでユーザーにメッセージを表示することができます。
  Windows ストアのスプラッシュ スクリーンの拡張ページを次に示します。この画面は初期スプラッシュ スクリーンと同じですが、"不定リング" プログレス コントロールが追加してあり、アプリが読み込み中であることがユーザーに示されます。

不定リングなどのプログレス コントロールの概要については、「プログレス コントロールのガイドライン」をご覧ください。

ヒント フラグメント読み込みを使ってスプラッシュ スクリーンの拡張ページを読み込む場合、Windows のスプラッシュ スクリーンが閉じてから、拡張されたスプラッシュ スクリーン ページが表示されるまでの間に、ちらつきが生じることがあります。このちらつきが生じるのは、**activated** イベント ハンドラーの実行が終了する前に、フラグメント読み込みがスプラッシュ スクリーンの拡張ページの読み込みを非同期的に開始するためです。スプラッシュ画面のサンプルで示されている設計パターンを使うと、この不快なちらつきは生じません。スプラッシュ スクリーンの拡張ページを、フラグメントとして読み込む代わりに、単純にアプリの UI 上に描画します。追加の読み込みタスクが完了すると、スプラッシュ スクリーンの拡張ページの表示をやめてアプリのランディング ページを表示できるようになります。また、スプラッシュ スクリーンの拡張ページをフラグメントとして読み込む場合は、アクティブ化の保留を取得して **activated** イベントに非同期的に応答することで、ちらつきを防ぐこともできます。**activatedOperation.getDeferral** メソッドを呼び出して、activated イベントの保留を取得します。

- スプラッシュ スクリーンやスプラッシュ スクリーンの拡張ページを広告の表示に使わないでください。

  スプラッシュ スクリーンの目的は、アプリの読み込み中に、指定したアプリが適切に開始されていることをユーザーに知らせることです。関係ない要素をスプラッ

シュ スクリーンに含めると、アプリをひとめで識別することが難しくなり、適切なアプリが起動されていないのではないかとユーザーを不安にさせます。

- 複数の異なるスプラッシュ スクリーン画像を表示するメカニズムとして、スプラッシュ スクリーンの拡張ページを使わないでください。
  スプラッシュ スクリーンやスプラッシュ スクリーンの拡張ページの目的は、ユーザーにスムーズで洗練された読み込みエクスペリエンスを提供することです。スプラッシュ スクリーンの拡張ページを使って複数の異なるスプラッシュ スクリーン画像を表示することは、この目的から離れた行為であり、ユーザーに不快な印象を与えたり、混乱を招く可能性があります。スプラッシュ スクリーンの拡張ページは、他のタスクが完了するまで、現在の読み込みエクスペリエンスを続けるためだけに使ってください。
- スプラッシュ スクリーンやスプラッシュ スクリーンの拡張ページをバージョン情報の表示に使わないでください。
  スプラッシュ スクリーンを、バージョン情報やその他のアプリ メタデータの表示に使わないでください。これらの情報は、アプリの Windows ストアの説明や、アプリ自体の中に表示します。

ユーザー エクスペリエンス

- アプリを明確に特徴付ける画像を使います。アプリを明確に特徴付ける画像と配色を使って、ユーザーが正しいアプリを起動したことを確信できるようにします。独特な画面にすると、ブランドを印象付ける効果もあります。
- ビジュアル効果を高めるため、スプラッシュ スクリーンの画像には透過的な PNG を使います。透過的な PNG を使うと、スプラッシュ スクリーンの画像全体に選択した背景色が適用されます。透過的でないと、画像の背景色が異なる場合に、スプラッシュ スクリーンがちぐはぐで魅力のないものになる可能性があります。
- 3 つの各倍率に合わせたサイズで、スプラッシュ スクリーンの画像のバージョンを用意します。すべてのアプリのスプラッシュ スクリーンの画像は、デバイスが等倍表示の場合、620 x 300 ピクセルにする必要があります。また、1.4 倍と 1.8 倍の追加のスプラッシュ スクリーン画像も含めることをお勧めします。3 つの倍率の各画像を用意すると、異なるデバイスでも、きれいで統一感のある起動エクスペリエンスを提供できます。

各倍率で必要なスプラッシュ スクリーン画像のサイズを判断するには、次の表をご覧ください。

スケール	画像サイズ (ピクセル)
等倍	620 x 300
1.4 倍	868 x 420
1.8 倍	1116 x 540

- システムによってスプラッシュ スクリーン画像に割り当てられた領域を使う画像を選びます。スプラッシュ スクリーン画像を選ぶときには、各倍率で割り当てられた領域を活用するようにします。各倍率でのスプラッシュ スクリーン画像のサイズを判断するには、倍率と画像 サイズの表をご覧ください。
  こうすることで、画像の品質を確保し、高品質なスプラッシュ スクリーンを作成できます。
- スプラッシュ スクリーンが消えた後、システムとイベントに関連する UI を表示します。スプラッシュ スクリーンの **dismissed** イベントをリッスンすることで、システムやイベントに関連する UI を安全に表示するタイミングを識別できます。そうしないと、スプラッシュ スクリーンが消える前に、関連付けられた UI (検索ウィンドウ、メッセージ ダイアログ、Web 認証ブローカーなど) が表示されてしまう場合があります。その場合、期待とは異なるビジュアル効果が発生してしまいます。
- スプラッシュ スクリーンが消えた後、導入アニメーションを開始します。多くのアプリは、アプリのランディング ページが読み込まれるたびに、コンテンツの導入アニメーションを表示しようとします。スプラッシュ スクリーンの **dismissed** イベントをリッスンして、アニメーションを開始するタイミングを識別できます。

スプラッシュ スクリーンの拡張ページ

- スプラッシュ スクリーンの拡張ページの外観は、Windows で表示されるスプラッシュ スクリーンに似せるようにしてください。スプラッシュ スクリーンの拡張ページでは、Windows のスプラッシュ スクリーンと同じ背景色と画像を使う必要があります。一貫性のある画像と背景色を使うと、Windows のスプラッシュ スクリ

ーンからアプリのスプラッシュ スクリーンの拡張ページへの遷移が洗練され、ユーザーに不快感を与えずにすみます。

- スプラッシュ スクリーンの拡張ページの画像を、Windows がスプラッシュ スクリーン画像を表示するときの座標に配置します。

  SplashScreen クラスを使ってスプラッシュ スクリーンの拡張ページの画像を配置する方法については、「スプラッシュ画面を拡張する方法」をご覧ください。
- スプラッシュ スクリーンの拡張ページの画像の位置を調整して、スナップや回転などのサイズ変更イベントに応答します。スプラッシュ スクリーンの拡張ページは、**onresize** イベントをリッスンして、アプリがスナップされたりデバイスが回転したりした場合にスプラッシュ スクリーンの画像の座標に合わせて調整する必要があります。こうすることで、ユーザーが画面上でどのようなデバイスの操作やアプリのレイアウト変更をしていても、アプリの読み込みエクスペリエンスをスムーズで洗練されたものにすることができます。
- スプラッシュ スクリーンの拡張ページを数秒以上表示する場合は、アプリがまだ読み込みを行っていることがユーザーにわかるように、進行状況リングを追加します。進行状況不定リング コントロールを使うと、アプリがクラッシュしたのではなく、間もなく準備ができることをユーザーに知らせることができます。アプリがユーザーのために何をしているかを簡潔に説明する 1 行のテキストを、進行状況リングと一緒に表示することを検討してください。たとえば、スプラッシュ スクリーンの拡張ページには、進行状況リングと、読み込み中であることを示すメッセージを追加できます。

  アプリの応答性が高いと感じさせ、ユーザーに情報を提供し続けることは、ユーザーの読み込みエクスペリエンスに対する評価を高めるための優れた方法です。進行状況不定リングとテキストを追加する方法については、「クイック スタート: プログレス コントロールの追加」をご覧ください。

その他の使い方のガイダンス

すべての Windows ストア アプリにはスプラッシュ スクリーンが必要です。スプラッシュ スクリーンは、スプラッシュ スクリーン画像と背景色で構成されています。 その両方をカスタマイズできます。

Windows では、ユーザーがアプリを起動すると、即座にこのスプラッシュ スクリーンが表示されます。これによって、アプリ リソースの初期化中であることがユーザーに示されます。 スプラッシュ スクリーンは、アプリが操作できる状態になるとすぐに、Windows によって閉じられます。

スプラッシュ スクリーンが適切にデザインされていると、アプリがより魅力的なものになります。Windows ストアでは、以下のような装飾の少ないシンプルなスプラッシュ スクリーンを使っています。

以下のスプラッシュ スクリーンは、緑の背景色と透過的な PNG を組み合わせて作られています。

SplashScreen クラスを使って、スプラッシュ スクリーンを拡張し、導入アニメーションをトリガーすることで、アプリの起動エクスペリエンスをカスタマイズできます。

トラブルシューティング

JavaScript: スプラッシュ スクリーン拡張ページへの遷移中のちらつきの回避

「スプラッシュ画面を拡張する方法」には、ちらつきが生じるのを回避するためのアドバイスを記載しています。スプラッシュ スクリーンの拡張ページへの切り替え中にちらつきが生じる場合は、<img> タグに onload="" を追加して、 のようにします。こうすることで、スプラッシュ スクリーンの拡張ページへの切り替え前に画像がレンダリングされるまでシステムを待機させて、ちらつきを防ぐことができます。

C#: スプラッシュ スクリーンの拡張ページへの遷移中のちらつきの回避

「スプラッシュ画面を拡張する方法」の手順に従った場合、スプラッシュ スクリーンの拡張ページへの切り替え中にちらつきが生じることがあります。このちらつきは、ページ コンテンツのレンダリングが終わる前に (Window.Current.Activate を呼び出して) 現在のウィンドウをアクティブ化する場合に生じます。現在のウィンドウをアクティブ化する前にスプラッシュ スクリーンの拡張ページの画像が読み取られていることを確認することで、ちらつきが生じる可能性を減らすことができます。さらにちらつきを防ぐには、現在のウィンドウをアクティブ化する前に、タイマーを使って、少しの間 (たとえば 50 ミリ秒) アプリを待機させる必要があります。残念ながら、ちらつきを防ぐ確実な方法は存在しません。XAML

は非同期的にコンテンツをレンダリングするため、レンダリングが完了するときを確実に予測する方法が存在しないからです。

「スプラッシュ画面を拡張する方法」の手順に従った結果、スプラッシュ スクリーンの拡張ページへの切り替え中にちらつきが生じる場合は、現在のウィンドウをアクティブ化する前にアプリを少し待機させ、スプラッシュ スクリーンの拡張ページの画像の読み取りが終わるようにするために、次の手順を実行します。

1. ExtendedSplash.xaml で、スプラッシュ スクリーンの拡張ページの画像のマークアップを更新して、スプラッシュ スクリーンの拡張ページの画像が読み取られたときにユーザーに通知します。

```
<Image x:Name="extendedSplashImage" Source="Assets/SplashScreen.png"
ImageOpened="extendedSplashImage_ImageOpened"/>
```

画像が読み取られた後、**ImageOpened** イベントが発生します。次の例に示すように、**ImageOpened** 属性を追加し、イベント ハンドラー名 (extendedSplashImage_ImageOpened) を指定して、**ImageOpened** イベントを登録する必要があります。

2. ExtendedSplash.xaml では、スプラッシュ スクリーンの拡張ページの画像が読み取られた後で、タイマーに基づいて現在のウィンドウをアクティブ化するコードを ExtendedSplash クラスに追加します。

```
        private DispatcherTimer showWindowTimer;
        private void OnShowWindowTimer(object sender, object e)
        {
            showWindowTimer.Stop();

            // Activate/show the window, now that the splash image has rendered
            Window.Current.Activate();
        }

        private void extendedSplashImage_ImageOpened(object sender,
    RoutedEventArgs e)
        {
            // ImageOpened means the file has been read, but the image hasn't
    been painted yet.
            // Start a short timer to give the image a chance to render, before
    showing the window
            // and starting the animation.
            showWindowTimer = new DispatcherTimer();
            showWindowTimer.Interval = TimeSpan.FromMilliseconds(50);
```

```
            showWindowTimer.Tick += OnShowWindowTimer;
            showWindowTimer.Start();
        }
```

この例は、**ImageOpened** イベントに応答する方法と、タイマーを使って、現在のウィンドウをアクティブ化する前にアプリを少しの間待機させる方法を示しています。

3. **OnLaunched** メソッドを次のように変更します。

```
        protected override void OnLaunched(LaunchActivatedEventArgs args)
        {
            if (args.PreviousExecutionState != ApplicationExecutionState.Running)
            {
                bool loadState = (args.PreviousExecutionState ==
        ApplicationExecutionState.Terminated);
                ExtendedSplash extendedSplash = new ExtendedSplash(args.SplashScreen,
        loadState);
                Window.Current.Content = extendedSplash;
            }

            // ExtendedSplash will activate the window when its initial content has been
        painted.
}
```

この例では、現在のウィンドウに対する **Activate** の呼び出しを削除しました。代わりにアクティブ化は、**ImageOpened** イベントが発生し、スプラッシュ スクリーンの拡張ページの画像が読み取られたことが示されてから、ExtendedSplash オブジェクトによって実行されます。

4. こうしたコードで実際にちらつきを回避できることを確認するために、できるだけ多くの種類のデバイスや状況下でコードをテストしてください。

タイルと通知のガイドライン

このセクションのガイドラインを使って、タイルや通知を使った、ユーザーに合わせた魅力的なエクスペリエンスの作成方法を学習します。

このセクションの内容

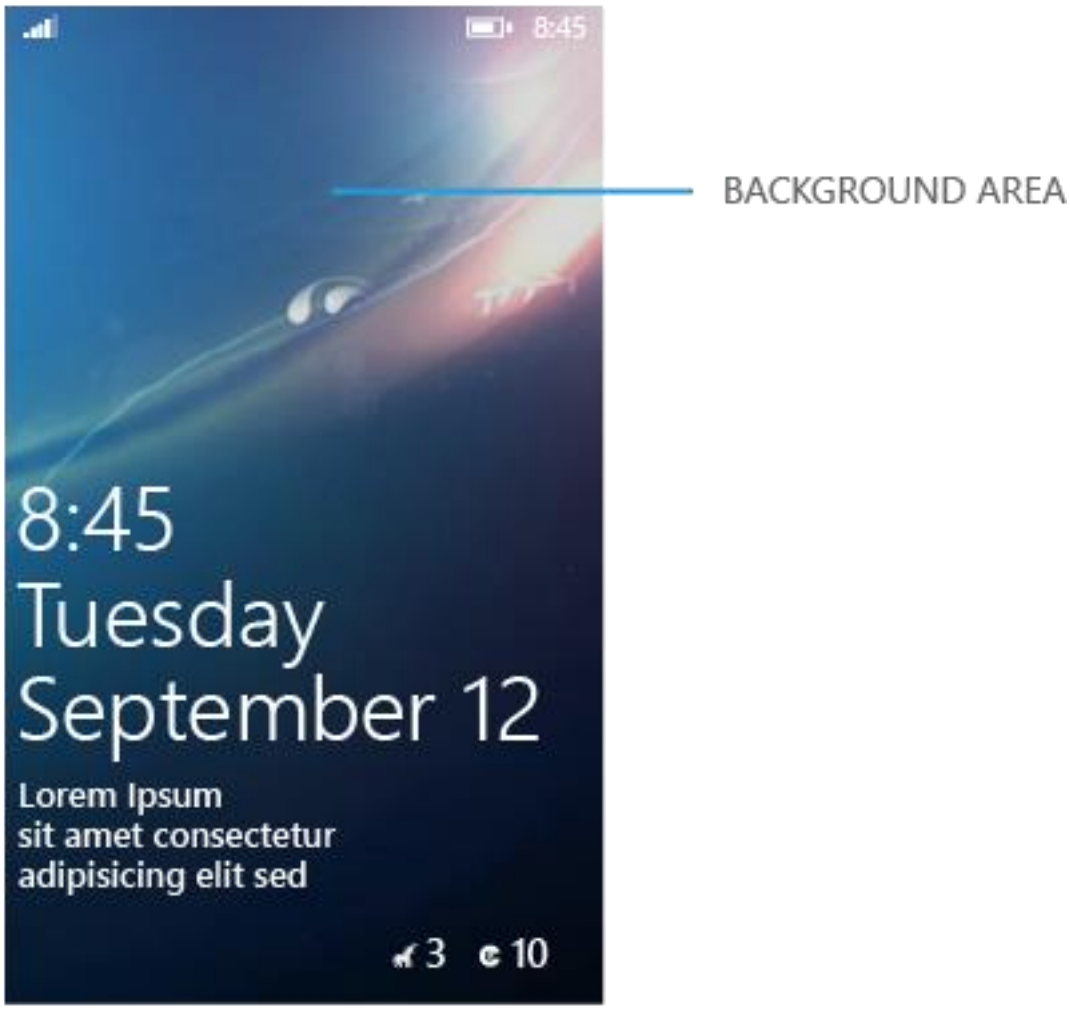
トピック	説明
Windows Phone のロック画面	ユーザーは、Windows Phone がロックされているときの背景プロバイダーとしてアプリを使うようにロック画面をカスタマイズできます。また、ロック画面に表示される基本的な通知 (メールやテキストなど) を、アプリで提供される通知を反映するように変更することもできます。このトピックでは、ロック画面の背景と通知の実装に関するベスト プラクティスについて説明します。
定期的な通知	このトピックでは、Windows ストア アプリで定期的な (ポーリングされた) 通知を使う際のガイドラインを示します。
プッシュ通知	このトピックでは、Windows ストア アプリでプッシュ通知を使う際の一般的なガイドラインとコーディングのガイドラインを示します。
直接通知	このガイドラインでは、有効な直接プッシュ通知を作成する方法について説明します。
スケジュールされた通知	Windows ストア アプリにスケジュールされたタイル通知やトースト通知を追加するときには、次のガイドラインに従ってください。
タイルとバッジ	ここでは、スタート画面とロック画面の両方における、アプリのタイルの作成と更新に関するベスト プラクティスとグローバリゼーション/ローカライズの推奨事項について説明します。また、アプリが Windows ストアで承認されるために満たす必要のあるタイルに関連する特別な要件も示します。
セカンダリ タイル	Windows ストア アプリでセカンダリ タイルを有効にし、関連する UI を設計するときは、次のガイドラインを考慮してください。
トースト通知	このトピックでは、トースト通知の用途を説明し、トースト通知の作成方法と送信方法について推奨事項を示します。

ロック画面の設計ガイドライン(Windows Phone)

ユーザーは、Windows Phone がロックされているときの背景プロバイダーとしてアプリを使うようにロック画面をカスタマイズできます。また、ロック画面に表示される基本的な通知 (メールやテキストなど) を、アプリで提供される通知を反映するように変更することもできます。このトピックでは、ロック画面の背景と通知の実装に関するベスト プラクティスについて説明します。

ロック画面の背景

次に示すように、ロック画面の背景は、Windows Phone がロックされているときに表示される静止画像です。



通常、ユーザーは見た目が美しい画像や親しみのある画像をロック画面の背景として選びます。地味な画像や親しみにくい画像は、ユーザーはおそらく選びません。

背景画像を実装する場合は、次の基本事項に留意してください。

- 画面のテキストと調和するよう、画像内のテキストの量を減らします。
- 複雑にならないようにします。"煩雑な" 写真やアートを使うと、画面のテキストや通知が読みにくくなります。

以下の画像では、ロック画面の背景画像の一部としてテキストまたはロゴを使うためのサイズ設定情報を示します。

WVGA (480 x 800 ピクセル)

WXGA (768 x 1280 ピクセル)

HD720p (720 x 1280 ピクセル)

デザインに関するその他の考慮事項

次の画像は、優れたロック画面デザインの 3 つの例を示します。

ロック画面: ロゴとテキスト、テキストのみ、ロゴのみ

- 画面上の日付、時刻、または通知と競合しないよう、画面上のロゴとテキストは小さなサイズを維持します。
- ロゴを追加する場合は、多少透明にすることを検討します。
- テキストを追加する場合は、画像に直接関連のある内容である必要があります。
- ロック画面イメージのビジュアルな焦点は、ロゴやテキストではなく、画像に合わせる必要があります。

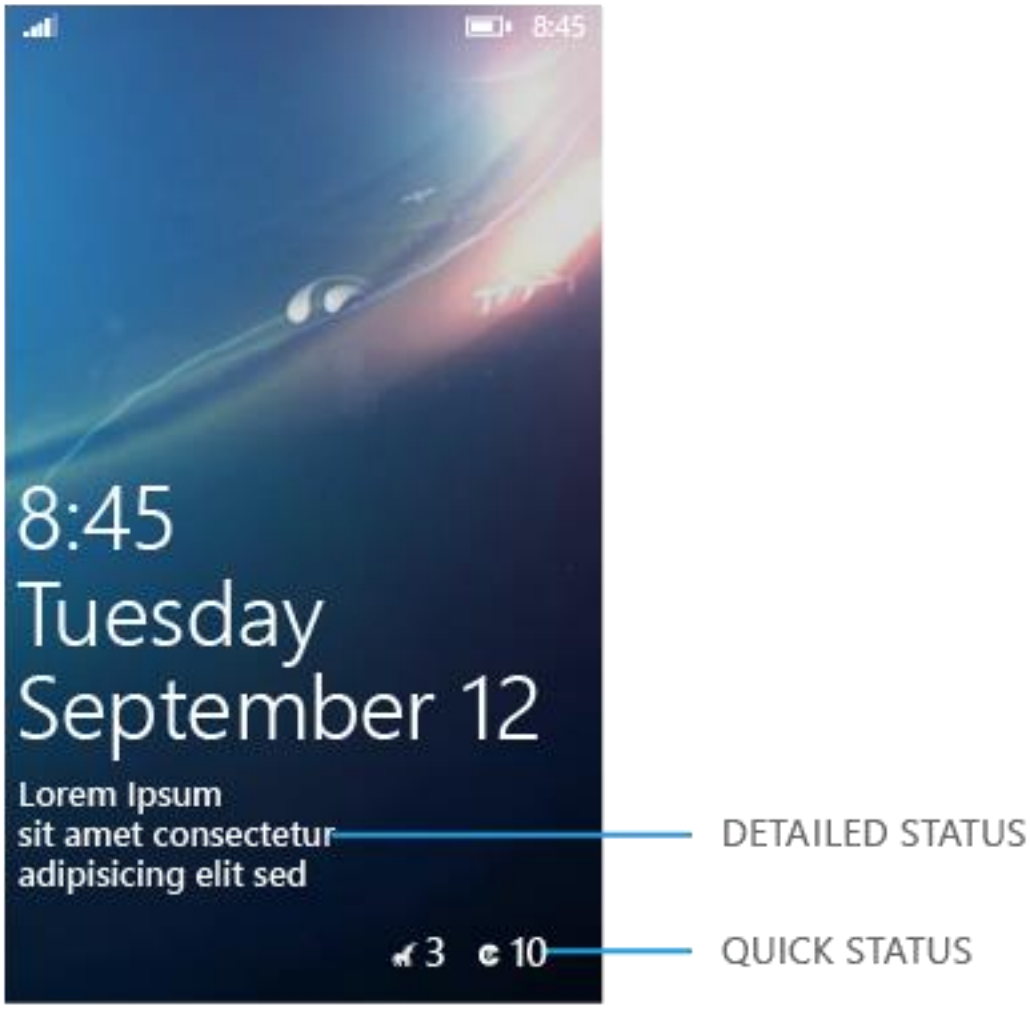
注 エラー発生時にロック画面のビジュアルが表示されるように、既定の背景画像を用意してください。

ロック画面の背景を更新する

アプリからロック画面の背景を更新する方法について詳しくは、「Windows Phone 8 のロック画面の背景」をご覧ください。

ロック画面の通知

次の画像には、詳細ステータスと簡易ステータスという 2 種類のロック画面の通知が示されています。各アプリのテキスト、アイコン、カウントは、アプリのプライマリ タイルから直接取得されます。



次の画像には、詳細ステータスと簡易ステータスという 2 種類のロック画面の通知が示されています。各アプリのテキスト、アイコン、カウントは、アプリのプライマリ タイルから直接取得されます。

- **詳細ステータス:** 通知テキストは白いピクセルで表示されます。表示される説明は、プライマリ タイルのものが使われます。既定のタイルがフリップ タイルの場

合、ワイド タイルの情報がロック画面の通知に表示されます。ただし、ワイド タイルの情報がない場合は、普通サイズのタイルの情報が表示されます。

- **簡易ステータス:** 開発者が用意したアイコンによって、ロック画面の通知領域でアプリが識別されます。アイコン イメージには、白のピクセルと、透かしと呼ばれるある程度の透明度のみが含まれる必要があります。実装した場合、プライマリ タイルのカウントもアイコンの横に表示されます。

ロック画面の通知領域にアプリを追加する

ロック画面の通知領域にアプリを追加する方法について詳しくは、「Windows Phone 8 のロック画面の通知」をご覧ください。

定期的な通知のガイドライン

製品通知は、クラウド サービスをポーリングして新しいコンテンツの有無を調べて、タイルとバッジを一定の間隔で更新します。Windows では、各ポーリング間隔の開始時にサービスに要求を送り、サービスが返すコンテンツをダウンロードして、アプリのタイルに新しいコンテンツを表示します。 詳しくは、「定期的な通知の概要」をご覧ください。このトピックでは、Windows ストア アプリで定期的な (ポーリングされた) 通知を使う際のガイドラインを示します。

アプリに定期的な通知を含めるかどうか

定期的に一定間隔で更新する必要のあるコンテンツをアプリで提供している場合に、定期的な通知を使います。たとえば、この通知の種類は次のような場合に適しています。

- 現在の予報を表示するライブ タイルを 30 分ごとに更新する天気予報アプリ。
- 毎朝、新しい日々のキャンペーンをユーザーと共有するアプリ。

定期的な通知をトースト通知で使うことはできないことに注意してください。時間を争う差し迫ったアラート (ニュース速報の更新など) やトースト通知を使うスケジュールされたアラームを共有する場合は、プッシュ通知またはスケジュールされた通知を使います。4 種類

の利用できる通知オプション (ローカル、スケジュール、プッシュ、定期的) を比較するには、「通知配信方法の選択」をご覧ください。

推奨と非推奨

全般

- 定期的な通知は、該当しなくなったら期限切れにします。たとえば、真夜中に終了する特別なオンライン オファーは、期限が切れた後、表示してはいけません。
- サーバーに更新を要求する頻度は、30 分ごとに 1 回を超えないようにします。この間隔であれば、ユーザーに負担をかけずにタイルの状態を最新に保つことができます。
- 通知コンテンツは、ホーム ページやランディング ページなど、アプリ内の目立つ場所に表示します。こうすると、ユーザーがタイル通知を受けてアプリを起動したときに、ユーザーの目をひいた通知のコンテンツを簡単に見つけることができます。
- ニュース速報など、ユーザーが即座に受け取ることを求めているコンテンツに定期的な更新を使わないでください。時間を争う更新を配布するには、プッシュ通知を使います。
- 定期的な通知を使って、ライブ タイルに広告を表示しないでください。タイルに広告を表示するのは禁止です。

効果的なタイルとバッジの設計に関する推奨事項については、「タイルとバッジのガイドライン」をご覧ください。

コード作成

- アプリを起動したりフォーカスを移動したりするたびに **StartPeriodicUpdate** または **StartPeriodicUpdateBatch** メソッドを呼び出します。これにより、アプリを起動したり切り替えたりするたびにタイルのコンテンツが更新されるようになります。
- クライアントのポーリング頻度に合わせて、Web サービスのタイルとバッジの XML コンテンツを更新します。たとえば、タイルが 30 分間隔でポーリングするよ

うに設定されている場合は、Web サービスのコンテンツも 30 分おきに更新します。

- クラウド サービスにアクセスできなくなった場合や、ユーザーがネットワークに長時間接続していない場合は、古いコンテンツや無効になったコンテンツをタイルから削除します。たとえば、深夜 0 時に期限が切れるショッピング キャンペーンの場合は、有効期限を深夜 0 時に設定します。有効期限の設定について詳しくは、「定期的な通知の概要」をご覧ください。
- 特定の時刻に更新を行うには、**StartPeriodicUpdate** または **StartPeriodicUpdateBatch** の startTime パラメーターを使います。startTime は、初回ポーリングの時刻のみを指定します。それを基に、後続のポーリングのタイミングが決まります。繰り返し間隔を 24 時間として startTime を 2:00 PM に設定した場合は、毎日午後 2 時になると更新が行われます。

注 タイルは、任意の時間に、最大 5 件の通知を順番に表示できます。キューに通知が 5 件ある場合、既定では、次の新しい通知によってキューの最も古い通知が置き換えられます。ただし、**StartPeriodicUpdateBatch** を使うと、サービス側で X-WNS-Tag という HTTP 応答ヘッダーを使って通知にタグを付けて、キューの置き換えポリシーを変更することができます。キュー内の既にある 5 件の通知とタグが合致する新しい通知を受け取ると、(最も古い通知が自動的に置き換えられる代わりに) タグが合致する古い方の通知が新しい通知で置き換えられます。タグと通知キューの使用について詳しくは、「ローカル通知で通知キューを使用する方法 (HTML)」または「ローカル通知で通知キューを使用する方法 (XAML)」をご覧ください。

プッシュ通知のガイドライン

プッシュ通知はクラウド サーバーから送られて、アプリのライブ タイルを更新すると共に、トースト通知を送ります。このトピックでは、Windows ストア アプリでプッシュ通知を使う際の一般的なガイドラインとコーディングのガイドラインを示します。

プッシュ通知を使うかどうか

プッシュ通知の配信方法では、アプリが実行されていなくても、いつでもユーザーがアプリから通知を受け取ることができます。

プッシュ通知は、アプリで次の情報を共有する場合に便利なオプションです:

- リアルタイム更新 (ゲーム内のスポーツ スコアなど)
- 予測不可能なタイミングで生成されるコンテンツ (ニュース速報、受信メール、ソーシャル ネットワークの更新など)

Windows ストア アプリで利用できる 4 種類の配信方法 (ローカル、スケジュール、プッシュ、定期的) の比較については、「通知配信方法の選択」をご覧ください。

推奨と非推奨

- タイル通知とトースト通知の全体的なガイドラインに従います。タイル通知またはトースト通知をローカルとクラウドのどちらで生成するかに関係なく、同じユーザー ガイドラインに従う必要があります。詳しくは、次のトピックをご覧ください。
  - タイルのガイドライン
  - トースト通知のガイドライン
- ユーザーのバッテリの寿命を考慮します。デバイスが低電力状態でも、ユーザーは通知をいつでも受け取ることができます。送る通知が増えると、必要なリソースが増え、デバイスのスリープ状態を解除する頻度も増えることになります。通知の頻度を決めるときにはこの点に注意してください。
- 快適なユーザー エクスペリエンスを実現できる最低限の通知頻度を選びます。通知の頻度を増やすことで、必ずしもアプリの価値が高まるわけではありません。た

とえば、タイルのコンテンツの更新頻度が多すぎると、一部の更新データをユーザーに見てもらえなくなります。

- プッシュ通知を使って機密データや重要なデータを送らないでください。たとえば、銀行の口座番号やパスワードを通知で送らないでください。
- 重要な通知に Windows プッシュ通知サービス (WNS) を使わないでください。WNS は信頼性の高いサービスですが、通知の配信は保証されていません。
- プッシュ通知を広告やスパムに使わないでください。WNS にはユーザーを保護する権利があります。アプリで通知を使う方法が不適切だと考えられる場合、WNS はそのアプリがプッシュ通知を利用できないようにブロックできます。ユーザーからアプリに悪意があるという報告があった場合、そのアプリは Windows ストアの削除ポリシーの対象となる可能性があります。

開発者向け

- WNS を使うことができるように、アプリをダッシュボードに登録します。アプリ サーバーでは、ダッシュボードで提供される特定の資格情報を使って認証を行い、通知を送る必要があります。
- アプリを起動するたびにチャネルを要求します。チャネルの URL は期限切れになる場合があり、URL を要求するたびに常に同じであることが保証されているわけではありません。返されたチャネルの URL がこれまで使っていた URL と異なる場合は、アプリ サーバーで参照を更新します。
- チャネルの URL が WNS から提供されたものであることを確認します。WNS 以外のサービスに通知をプッシュしようとしないでください。チャネルの URL で "notify.windows.com" (Windows または Windows Phone) か、"s.notify.live.net" (Windows Phone のみ) のいずれかのドメインが使われていることを確認します。
- アプリ サーバーへのチャネル登録のコールバックを常にセキュリティで保護します。アプリがチャネルの URL を受け取り、その URL をアプリ サーバーに送るときには、セキュリティで保護された方法でこの情報を送る必要があります。チャネルの URL の送受信に使うメカニズムには、認証と暗号化を適用します。
- アクセス トークンを再利用します。アクセス トークンを使って複数の通知を送ることができるため、サーバーでは通知を送るたびに再認証を行わなくて済むように、アクセス トークンをキャッシュする必要があります。トークンが期限切れに

なっている場合は、アプリ サーバーにエラーが返されます。この場合、アプリ サーバーの認証を行い、通知を再試行する必要があります。

- パッケージ セキュリティ識別子 (PKSID) と秘密鍵を第三者と共有しないでください。これらの資格情報は、セキュリティで保護された方法でアプリ サーバーに保存します。秘密鍵が侵害されていると思われる場合は、新しいキーを生成します。攻撃者の標的になりにくくするために、新しい秘密鍵を定期的に生成します。

直接通知のガイドライン

直接通知はプッシュ通知の一種ですが、他の 3 種類のプッシュ通知 (トースト、タイル、バッジ) と異なり、関連付けられた UI がありません。他のプッシュ通知と同様に、Windows プッシュ通知サービス (WNS) 機能は、クラウド サービスからアプリに直接通知を配信します。このガイドラインでは、有効な直接プッシュ通知を作成する方法について説明します。

直接通知を使うかどうか

直接通知は、ユーザーがアプリに権限を与えている場合にアプリによるバックグラウンド タスクの実行をトリガーするなど、さまざまな目的で使うことができます。直接通知はまた、アプリでクラウド サービスからダウンロードできるデータがある場合に通知を受け取る優れた方法でもあります。アプリは次のことができます。

- 直接通知を、ファイル ダウンロードを開始するトリガーとして使う。たとえば、ユーザーがネット上で電子ブックを購入する場合、そのユーザーのリーダー アプリに直接通知を送り、その新しい電子ブックのダウンロードをトリガーするというものです。
- 直接通知を使い、インスタント メッセージや電話の受信を通信アプリに知らせる。通知を受けた通信アプリは、接続を確立し、ローカル トースト通知を使ってユーザーに知らせることができます。
- 直接通知を使い、クライアントとクラウド サービス間の同期操作を調整する (リーダー アプリで最後に読まれたページの同期をトリガーするなど)。

アプリとの通信に Windows プッシュ通知サービス (WNS) を使うことで、固定ソケット接続の作成、HTTP GET メッセージの送信、サービスとアプリ間でのその他の接続などに伴う処理のオーバーヘッドを回避できます。

詳しくは、「直接通知の概要」をご覧ください。

推奨と非推奨

- 直接通知で送信する情報の量はできる限り抑えてください。WNS では直接通知で 5 KB を超えるデータを送信できないことに注意してください。
- 直接通知は、アプリがクラウド サービスからダウンロードできる情報を通知内に含めるのではなく、そのような情報があるということを知らせるために使います。
- 通知内のバイナリ データは、直接通知に含める前に base64 にエンコードしてください。これにより、転送中にコンテンツが不適切にエンコードされることがなくなり、クライアントによって確実に取得されます。
- 快適なユーザー エクスペリエンスを実現できる最低限の通知頻度を選びます。
- アプリを起動するたびにチャネルを要求します。チャネルの URL は期限切れになる場合があり、URL を要求するたびに常に同じであることが保証されているわけではありません。返されたチャネルの URL がこれまで使っていた URL と異なる場合は、アプリ サーバーで参照を更新します。
- チャネルの URL が WNS から提供されたものであることを確認する: WNS 以外のサービスに通知をプッシュしようとしないでください。チャネルの URL で "windows.com" ドメインが使われていることを確認します。
- アプリ サーバーへのチャネル登録のコールバックを常にセキュリティで保護します。アプリがチャネルの URL を受け取り、その URL をアプリ サーバーに送るときには、セキュリティで保護された方法でこの情報を送る必要があります。チャネルの URL の送受信に使うメカニズムには、認証と暗号化を適用します。
- アクセス トークンを再利用します。アクセス トークンを使って複数の通知を送ることができるため、サーバーでは通知を送るたびに再認証を行わなくて済むように、アクセス トークンをキャッシュする必要があります。トークンの有効期限が切れると、アプリ サーバーがエラーを受け取ります。アプリ サーバーを認証し、通知をもう一度試みます。

- 直接通知を利用して、連続した通知に情報を少量ずつ含める方法でアプリに情報をストリーミングしないでください。直接通知の送信は、クラウド サービスでトリガーされたイベントに応答する場合だけに限定する必要があります。
- バックグラウンド タスクの実行を継続させるという目的でクラウド サービスから直接通知を送信しないでください。このような処理は、ユーザーのバッテリ寿命を無駄に消費することになります。直接通知は、有用な情報をアプリに伝える必要があります。
- 関連付けられたバックグラウンド タスクによるリソース クォータの超過を引き起こす速度で直接通知を送らないでください。詳しくは、「バックグラウンド タスクのガイドライン」をご覧ください。
- 重要な通知の送信に WNS を使わないでください。WNS は信頼性の高いサービスですが、通知の配信は保証されていません。
- 通知を広告やスパムに使わないでください。WNS にはユーザーを保護する権利があります。アプリで通知を使う方法が不適切だと考えられる場合、WNS はそのアプリが通知を利用できないようにブロックできます。ユーザーからアプリに悪意があるという報告があった場合、そのアプリは Windows ストアの削除ポリシーの対象となる可能性があります。
- 直接通知にはサイズがゼロのペイロード コンテンツを含めないでください。ペイロードが含まれない直接通知は WNS によってドロップされ、アプリには配信されません。
- 直接通知を使って機密データや重要なデータを送らないでください。
- パッケージ セキュリティ識別子 (PKSID) と秘密鍵を第三者と共有しないでください。これらの資格情報は、セキュリティで保護された方法でアプリ サーバーに保存します。新しい秘密鍵を定期的に生成します。秘密鍵が侵害されている場合は、すぐに新しいキーを生成します。

その他の使い方のガイダンス

アプリで直接通知によってトリガーされるバックグラウンド タスクを使う前に、他の通信手段を利用できないか検討してください。ユーザーはアプリに対し、バックグラウンド タスクを実行するアクセス許可を明示的に付与する必要があります。このアクセス許可は、同時に最大 7 つのアプリに付与できます。アプリで他の通信メカニズム (標準のプッシュ通

知、トースト更新など) を使えば、ユーザーの許可がなくてもバックグラウンド タスクを実行できます。

バックグラウンド タスクに替わる手段として、次の方法を利用できます。

- ユーザーに知らせるには、トースト プッシュ通知を送信します。
- タイルを更新するには、タイル プッシュ通知を利用します。

スケジュールされた通知のガイドライン

スケジュールされた通知を使って、定期的にアプリのタイルを更新したり、ユーザーにトースト通知を送ったりすることができます。Windows ストア アプリにスケジュールされたタイル通知やトースト通知を追加するときには、次のガイドラインに従ってください。

アプリでスケジュールされた通知を使うかどうか

アプリ内からコンテンツを使ってアプリのタイル、バッジ、またはトースト通知を定期的に更新する場合は、スケジュールされた通知を使います。スケジュールされた通知は、タイルやバッジを更新する時刻またはトースト通知を表示する時刻を指定できる点を除いて、ローカル通知と同じです。

通知の内容が即時性を求める場合 (ニュース速報など)、予測不可能なタイミングで表示される場合 (受信メールなど)、アプリ外部からのデータに依存する場合、またはアプリが実行されていないときに更新する必要がある場合、別の形式の通知配信を使う必要があります。

Windows ストア アプリで利用できる 4 種類の通知配信の概要については、「通知配信方法の選択」をご覧ください。

推奨と非推奨

- タイル通知またはトースト通知の内容と各通知の更新頻度を決定する場合は、「タイルのガイドライン」と「トースト通知のガイドライン」の推奨事項に従います。
- **MaintenanceTrigger** クラスを使ってスケジュールを定期的に更新するために、**background tasks** を使うことを検討します。たとえば、最初に 1 週間の通知をあらかじめスケジュールしておき、**MaintenanceTrigger** クラスを使って以降の週のスケジュールを引き続き設定します。こうすると、ユーザーがある週にアプリを起動しなくても、継続的にスケジュールを設定できます。
- **timeZoneChange** システム トリガーを使って、システム クロックの変更 (夏時間など) に対応することを検討します。既定では、スケジュールされた通知は協定世界時 (UTC) でトリガーされます。システム クロックの変更に対応して自動的に更新されるわけではありません。たとえば、アラーム アプリでは、システム時刻が変更された場合にアラームの予定時刻を変更する必要があります。そのためには、アプリで **timeZoneChange** トリガーに応答してタイミングを適切に調整するバックグラウンド タスクを使います。

タイルとバッジのガイドライン

ここでは、スタート画面とロック画面の両方における、アプリのタイルの作成と更新に関するベスト プラクティスとグローバリゼーション/ローカライズの推奨事項について説明します。また、アプリが Windows ストアで承認されるために満たす必要のあるタイルに関連する特別な要件も示します。

推奨と非推奨

- 一般的なガイドライン
- 既定のタイル
- プレビュー テンプレート
- バッジ
- タイル通知

一般的なガイドライン

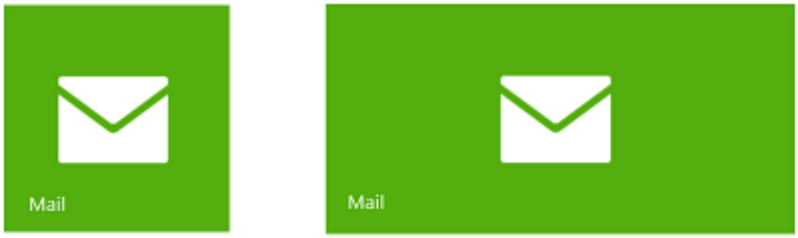
- アプリでタイル通知を使ってユーザーに更新を送らない場合は、小と普通サイズのタイルだけを使います。ワイドと大サイズのタイルのコンテンツは、定期的に更新し、新鮮な状態を保つ必要があります。ライブ タイルを使わない場合は、マニフェストでワイドまたは大サイズ ロゴを指定しないでください。
- アプリで、短い要約通知を使うシナリオだけをサポートする場合は、バッジと小または普通サイズのタイルだけを使います。要約通知とは、**badge image** または 1 つの数字でのみ表すことができる通知です。たとえば、SMS アプリで、受け取った新しいテキストの数だけを伝えるために通知を使う場合などが、このシナリオに該当します。マニフェストにワイド ロゴを指定しないでください。
- アプリでスタート画面に詳細を表示する必要のない更新データを送る場合は、小と普通サイズのタイルのみを使います。たとえば、給与明細書アプリでは、給与の金額などの詳細を説明しないで、新しい給与明細書が入手可能であることだけを示すことができます。マニフェストにワイドまたは大サイズ ロゴを指定しないでください。
- ユーザーに表示する新しい興味深いコンテンツがアプリにあり、その通知が頻繁に更新される (少なくとも週 1 回) 場合にのみ、ワイドまたは大サイズのタイルを使います。

- 1 つの通知で複数の記事を同時に表示したり、長い一覧を表示したり、ユーザーがスタート画面で大きなサイズで見たがるような画像を表示したりするために、大きいタイルを使います。
- 既定のタイル イメージは、アプリのブランドを表すために基本的にアプリのロゴのキャンバスとして使います。
- ユーザーに合わせた興味を引く新しいコンテンツがない場合は、ライブ タイルは使わないでください。たとえば、電卓アプリなどにはそのようなコンテンツはありません。
- 伝達する情報で興味を引くものがユーザーの最新状態だけの場合、ライブ タイルは使わないでください。ユーティリティ アプリ、Microsoft Visual Studio のような開発者ツール、ユーザーの最新セッションのサムネイルだけを表示するブラウザーでは、ライブ タイルを使わないでください。
- ユーザーへのスパムの送信や広告の表示にライブ タイルを使わないでください。Windows ストアから除外されることになります。
- ブランドを通知キューの項目の 1 つまたはプレビュー テンプレートのフレームの 1 つとして使わないでください。これら両方のシナリオでは、ユーザーの目に留まるよう、タイルへの変更をアニメーション化します。興味を引く新しいコンテンツを表示しないで、ブランドを表示するためだけにアニメーションを利用してユーザーの注意を引こうとしても、ユーザーを不快にするだけです。

既定のタイル

- ワイド ロゴを含めるか、ワイド ロゴと大サイズ ロゴの両方を含める場合は、指定する普通サイズ、ワイド、大サイズのタイル イメージとのデザインの関係を検討します。ユーザーはサポートされている任意のサイズでタイルを表示でき、いつでもそのサイズを変更できることを忘れないでください。ここでは、一般的な規則をいくつか示します。
  - タイルの横方向の中央に、ロゴを配置します。
  - 同じ高さの正方形タイルとワイド タイルの両方で、ロゴの縦方向の位置を同じに保ちます。大サイズ タイルで、ロゴの同じ比率の縦方向の位置を保ちます。
  - ロゴ イメージ自体にアプリ名が含まれていない場合は、タイルの下部にアプリ名を含めます。ただし、小サイズ タイルにはアプリ名を表示するオプションがありません。次の例は、両方のタイルの状態を示しています。

マニフェストに定義されているアプリ名の要素を使ったタイル



ロゴ イメージにアプリ名を含んだタイル

- 長い名前のアプリの場合、名前が 2 行に折り返される可能性があるため、ロゴ イメージと名前が重ならないことを確認します。たとえば、普通サイズとワイドのタイルでは、100% の画像リソースで約 80 x 80 ピクセルにロゴの大きさを制限するのが、確実な方法です。
- 画像内でロゴ自体の周りのスペースを透明にすると、Windows 8 の外観の一部として、アプリのブランドの色 (マニフェストで宣言済み) が、事前に割り当てられたグラデーションを伴い、透けて表示されます。この方法は、先に示したメール アプリのタイルなどのロゴで使われます。

- 既定のタイルは、アプリの起動を露骨に要求するようなテキストを含んだデザイン ("クリックしてください!" と表示されたタイルなど) にしないでください。
- ロゴにアプリ名を含める場合は、名前フィールドでアプリ名を繰り返さないでください。ここで示すように、どちらか一方のみを使ってください。

プレビュー テンプレート

- シナリオに、それぞれが独立しているイメージとテキスト コンテンツが含まれる場合は、プレビュー テンプレートを使います。たとえば、テンプレートの上部に旅行先の写真を表示し、下部に場所の名前を表示する場合があります。

- プレビュー テンプレートでアニメーションを表示すると、ユーザーの注意を引き付けます。よって、魅力的なコンテンツを提供するようにしてください。コンテンツが魅力的でないと、ユーザーを不快にするだけです。
- プレビュー テンプレートを使うと、表示をサイクルのいずれかの終わり (フレーム) (テキストが一番下にある状態またはテキストが一番上にある状態) で開始し、もう一方のフレームに上がっていくまたは下がっていくアニメーションを表示できます。したがって、各フレームのコンテンツが単独で動作できることを確認します。
- プレビュー テンプレートを使ってユーザーが既に知っていることに関する情報を表示しないでください。たとえば、ビデオが一時停止中であるという通知をタイルに表示する場合は、プレビュー テンプレートを使わないでください。
- 概念的にグループ化されていない通知には、プレビュー テンプレートを使わないでください。たとえば、写真がテキストとまったく関係がない場合は、プレビュー テンプレートを使わないでください。
- プレビュー アニメーションによって通知の最も重要な部分が画面から消される可能性がある場合は、プレビュー テンプレートを使わないでください。たとえば、気温と付随する画像 (笑っている太陽、または雲) を表示する天気予報アプリでは、プレビュー テンプレートを使うと、気温 (通知するポイント) がいつでも表示されるとは限りません。画像と気温を同時に表示する静的テンプレートの方が、ユーザーにとっては便利です。
- ニュース記事の場合のように、画像にコンテキストを与えるためにテキストが必要な場合は、プレビュー テンプレートを使わないでください。

バッジ

- アプリで要約通知を使うシナリオだけをサポートする場合は、バッジを表示する普通サイズのタイルだけをサポートします。たとえば、ショート メッセージ サービス (SMS) アプリで、受け取った新しいテキストの数だけを表示する場合などです。ユーザーがタイルを小サイズに変更しても、バッジは表示されることに注意してください。
- シナリオで伝える情報を小さな数値で表すことができる場合は、バッジに数値を表示します。バッジに常に 50 以上の数値を表示する可能性が高い場合は、システム グリフの使用を検討します。バッジの数値が大きくなりすぎないようにする方法と

して、絶対数ではなく、ユーザーがアプリを前回起動した時点からの数を示すことが挙げられます。たとえば、アプリがインストールされた時点からの不在着信の合計件数を示すよりも、ユーザーがアプリを前回起動した時点からの不在着信件数を示す方が実用的です。

- 数値が役に立たない場合や非常に大きくなる場合は、用意されているシステム グリフのいずれかを使って変化を示します。たとえば、大量の RSS フィードの新しい未読記事は膨大な数になる可能性があります。この場合、数値ではなく、newMessage システム グリフを使います。
- 数値に意味がない場合はグリフを使います。たとえば、タイルに再生リストの "一時停止" 通知を表示する場合、数値では意味がわからないため、paused グリフを使います。
- 数値では意味があいまいになる場合は、newMessage グリフを使います。たとえば、ソーシャル メディア タイルのバッジに "10" と示されている場合、10 個の新しいリクエスト、10 件の新しいメッセージ、10 個の新しい通知、これらの組み合わせなど、複数の解釈が可能となります。
- タイルのバッジに最大値の "99+" が常に表示される可能性のある大量シナリオ (メールやソーシャル メディアなど) では、newMessage グリフを使います。常に最大値が表示される可能性がきわめて高い場合、同じ数値が表示された状態が続くことになり、ユーザーにとって役に立たない情報を伝えることになります。
- ワイド タイルの本文コンテンツとは別の場所でバッジの数値を繰り返さないでください。これは、2 つのインスタンスにずれが生じる場合があるためです。
- グリフでユーザーに伝える内容が変わることがない場合は、グリフを使わないでください。グリフは通知や過渡的な状態を表します。永続的なブランド情報や状態を表すものではありません。

タイル通知

- ユーザーに関する情報を基に、タイルを使ってそれぞれのユーザーに合わせた通知を送ります。タイル通知は、対象のユーザーに関連している必要があります。利用する必要がある情報の大部分は特定のアプリの内部にあり、ユーザーのプライバシー選択によって制限される場合があります。たとえば、テレビ ストリーミング サービスでは、最も視聴率の高い番組に関する最新情報をユーザーに示すことができ

ます。また、交通情報アプリでは、ユーザーの現在の場所に基づいて (ユーザーが現在の場所を知らせることを許可した場合) 最適な地図を表示できます。

- アプリが接続され、最新のライブ コンテンツを受け取っているとユーザーが感じるように、タイルに更新データを頻繁に送ります。タイル通知の更新間隔は、特定のアプリのシナリオによって異なります。たとえば、活発に利用されるソーシャル メディア アプリの場合は 15 分おきに更新し、天気予報アプリは 2 時間おき、ニュース アプリは 1 日に数回、毎日情報を提供するアプリは 1 日に 1 回、雑誌アプリは月に 1 回更新します。アプリでの更新が週に 1 回未満の場合は、古いコンテンツが表示されないように、シンプルな普通サイズ タイルとバッジを併用することを検討します。
- ユーザーが十分な情報に基づいてアプリの起動が必要であるかを判断できるように、魅力的で役立つタイル通知を提供します。一般に、通知とは詳細を知るために、またはアクションを実行するためにアプリを起動するようユーザーを促すものです。たとえば、通知によってユーザーはソーシャル メディアの投稿に返信したり、ニュース記事の全文を読んだり、セールの詳細を知ったりする場合があります。
- アプリのホーム ページまたはランディング ページでホストされるコンテンツについての通知を送信します。このようにして、ユーザーは通知に応答してアプリを起動するときに、通知の対象となったコンテンツを簡単に見つけることができます。

その他の使い方のガイダンス

- タイルの設計哲学
- さまざまなタイル サイズの選択
- 既定タイルの使用
- プレビュー テンプレートの使用
- デザインに関するその他の考慮事項
- タイル通知の更新

タイルは、スタート画面でのアプリの表示です。タイルにより、アプリが実行されていないときでも、スタート画面に魅力的なコンテンツを表示できます。タイルをタップまたはクリックすると、アプリが起動します。タイルには、3 つの正方形サイズ (小、普通、大) と 1

つのワイド サイズがあります。普通、ワイド、大サイズに対して複数のテンプレートのバリエーションがあり、テキスト、画像、またはテキストと画像の組み合わせを使うことができます。"プレビュー テンプレート" と呼ばれる一部のテンプレートは、タイルのスペース内をスクロールして移動できる、2 つの積み重ねられたフレームで構成されています。プレビュー テンプレートは、普通サイズとワイド サイズのタイルで利用できます。

タイルをライブにするか (通知による更新)、または静的のままにすることができます。最初に使うタイルは既定のタイルで、アプリのマニフェストで定義されます。静的なタイルには、必ず既定のコンテンツ (一般に、タイル全体を占めるロゴ イメージ) が表示されます。ライブ タイルは既定のタイルを更新して新しいコンテンツを表示できますが、更新が期限切れになるか、または削除された場合に、既定の状態に戻すことができます。タイルには、数字またはグリフで表すことができる状態バッジも表示できます。

普通サイズのタイル、ワイド タイル、大きいタイルでは、オプションで下隅にアプリ名 (既定のタイルまたはライブ タイル) または小さいアイコン (ライブ タイルのみ) でブランドを表示できます。

以下に、常に注意すべき非常に重要なポイントを 2 つ示します。

- ユーザーはタイルのサイズを、タイルでサポートされている任意のサイズに変更できます。ユーザーのスタート画面で現在どのサイズが表示されているかを知る方法はありません。すべてのタイルで小と普通サイズをサポートする必要がありますが、オプションでワイドと大サイズもサポートできます。大サイズをサポートする場合は、ワイド サイズもサポートする必要があるので、大サイズをサポートするには 4 つのすべてのタイル サイズをサポートしなければならないことになります。大きいタイルとワイド タイルは、タイルでライブ更新をサポートする場合にのみ使うようにしてください。
- タイルがライブ タイルをサポートしている場合、ユーザーはタイル通知のオンとオフをいつでも切り替えることができます。タイル通知がオフの場合、タイルは静的です。

タイルの設計哲学

目標は、アプリの魅力的なタイルを作成することです。ライブ タイルを使う場合は、アプリの起動を促す訴求力があり、かつユーザーがスタート画面に表示する価値があると思うような魅力的で新しいコンテンツを提供することが目標です。そのためには、派手な色を使いすぎないようにします。騒ぎ立てる子供のように、注意を引くために派手に飾り立てたタイルよりも、シンプルで見やすい洗練されたデザインのタイルの方が成功します。

アプリを設計するときに、わざわざライブ タイルを開発する価値があるかどうか、疑問に感じるかもしれません。その理由はいくつかあります。

- タイルはアプリへの "入り口" です。魅力的なライブ タイルは、アプリが実行されていないときに、ユーザーを引き付けることができます。ユーザーが頻繁に使うアプリを重視する傾向は強まりつつあります。
- ライブ タイルは、Windows ストアの他のアプリ (ユーザーは、似たようなアプリでも、静的なタイルを持つものよりも、便利なライブ タイルを持つものを好む傾向にあります) や、ホーム画面に静的なタイルとアイコンしか表示できないオペレーティング システムのアプリとの差別化につながるセールス ポイントです。
- ユーザーがライブ タイルを気に入った場合、スタート画面にその目立つタイルがあると、アプリを繰り返し使ってもらえます。そのタイルから素敵なアプリ コンテンツを偶然に発見すると、ユーザーは楽しい気持ちになるでしょう。
- ライブ タイルを使うと、ライブ更新を見られるように、ユーザーがアプリをアプリ ビューからスタート画面にピン留めする可能性が高くなります。
- ユーザーがタイルを気に入らなければ、スタート画面の一番後ろに置く、ピン留めを外す、更新を無効にする、アプリをアンインストールするといった結果になりかねません。

次のような特性がライブ タイルの魅力を高めます。

- 頻繁に更新される新鮮なコンテンツ。アプリが実行されていなくても、ユーザーはアプリがアクティブだと感じます。
  例: 最新のヘッドラインや、新しいメールの数の表示。
- ユーザーに関する情報 (アプリの設定を通じてユーザーが設定できるようにした興味の対象など) に基づく、カスタマイズされた更新。
  例: ユーザーの趣味に合わせてカスタマイズされた、1 日限定キャンペーン。

- ユーザーの現在の状況に合ったコンテンツ。
  例: ユーザーの現在位置を使って該当する交通情報地図を表示する、交通情報アプリ。

さまざまなタイル サイズの選択

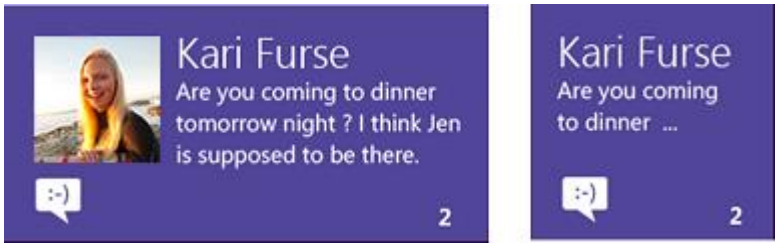
アプリには常に小サイズと普通サイズのタイルを用意する必要があります。アプリのマニフェストに、少なくとも普通サイズ タイルの画像アセットを用意する必要があります。小サイズ タイルのアセットも用意できますが、用意しない場合は、普通サイズ タイルのアセットの縮小バージョンが使われます。

また、ワイドまたは大サイズ タイルでも同様にするかどうかも決める必要があります。

- ワイド タイルをサポートするには、アプリのマニフェストに、既定のタイルの一部としてワイド (wide310x150) ロゴ イメージを含めます。その既定のワイド ロゴ イメージを含めなかった場合は、小 (square70x70) と普通 (square150x150) サイズだけがサポートされます。ユーザーがワイド サイズに変えることはできず、ワイド通知を受け取ることもできません。
- 大サイズ (square310x310) タイルをサポートするには、アプリのマニフェストに、既定のタイルの一部としてワイド ロゴ イメージと大サイズ ロゴ イメージを含めます。その既定の大サイズ ロゴ イメージを含めなかった場合は、ユーザーがタイルを大サイズに変えることはできず、大サイズ テンプレートを使う通知を受け取ることもできません。大サイズのタイルをサポートするにはワイド タイルのサポートが必要なので、既定の大サイズ ロゴ イメージを含めても既定のワイド ロゴ イメージを含めないと、両方を含めないのと同じ結果になります。

アプリが現在サポートしているタイル サイズ以外のサイズもサポートするには、追加の既定ロゴ イメージを含む更新されたマニフェスト付きでアプリの新しいバージョンをリリースする必要があります。

- 普通サイズ タイルに表示されるコンテンツは、ワイドや大サイズ タイルの場合よりも少なくなるので、コンテンツに優先順位を付けてください。ワイド タイルに表示できるコンテンツをすべて普通サイズ タイルに収めようとしないでください。さらに小さい小サイズ タイルでサポートされている唯一のライブ コンテンツは、バッジ通知です。



ワイド タイルのコンテンツに画像とテキストを含む場合は、正方形のプレビュー テンプレートを使って、コンテンツを 2 つのフレームに分割できます。ただし、画像自体で記事の概要を十分に伝えられない場合は、プレビュー テンプレートを使わないでください。

通知は、現在のタイルのサイズを知ることができないため、小サイズ タイル以外のサポートされているすべてのタイル サイズのテンプレート コンテンツを用意する必要があります。ワイド テンプレートのみを使って通知を定義し、タイルを普通サイズに表示している場合や、普通サイズ テンプレートのみを使って通知を定義し、タイルをワイドで表示している場合には、通知が表示されません。

規定のタイルの使用

アプリの既定のタイルは、アプリのマニフェストに定義されています。既定のタイルは静的で、一般的にはシンプルなデザインとなっています。アプリによっては、既定のタイルを使えば十分な場合があります。アプリのインストール後、ユーザーがタイルをアプリ ビューからスタイル画面にピン留めすると、既定のタイルが、通知を受け取るまでスタート画面に表示されます。ワイド ロゴ イメージを用意する場合は、タイルをスタート画面にピン留めする際の初期サイズを普通にするかワイドにするかを指定できます。マニフェストで指定されているワイド ロゴ画像によりワイド タイル サイズがアプリでサポートされている場合、アプリのタイルは既定でワイド タイルとしてピン留めされます。それ以外の場合、タイルは普通サイズでピン留めされます。ユーザーは、ピン留めされたタイルのサイズを、サポートされている任意のサイズに変更できます。期限切れになっていない新しい通知がなく、表示する必要がない場合は、ライブ タイルは既定の状態に戻ることがあります。

プレビュー テンプレートの使用

プレビュー テンプレートは、タイルのスペース内で 2 つのフレーム間の情報を切り替えるタイルのコンテンツを提供します。上のフレームは画像またはイメージ コレクションで、

下のフレームはテキストまたはテキストと画像です。例については、「タイル テンプレート カタログ」をご覧ください。

デザインに関するその他の考慮事項

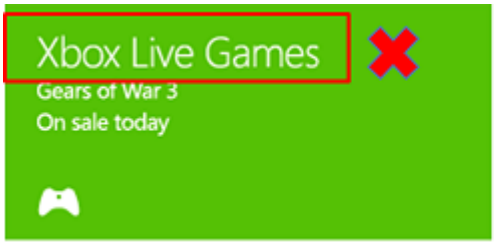
- タイルでアプリのブランド情報を伝える方法を決めるときは、次に示すようにアプリの名前を選びます。

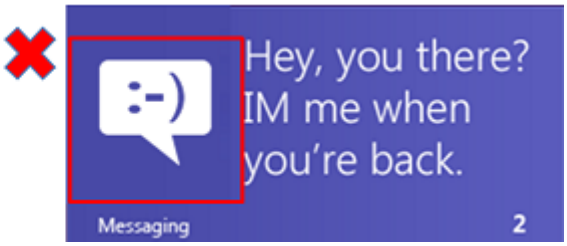
または、次に示すようにロゴ イメージを選びます。

これらの項目は最初はアプリ マニフェストで定義されます。開発者は、以降の各通知で 2 つのうちのどちらを表示するかを選ぶことができます。ただし、名前またはロゴを選んだら、常にそれを使って一貫性を保つ必要があります。スペースの制約により、一部のテンプレートでは名前を表示できず、ロゴを表示するか非表示にするかのオプションしかないことに注意してください。

- 画像要素またはテキスト要素を使って、タイルの通知にアプリのブランド情報を表示しないでください。ユーザーに対してアプリのブランドを印象付け、一貫性を保つには、アプリ名 (短い名前) またはロゴ イメージなど、その目的のために提供されたテンプレートの要素を使ってブランドを表す必要があります。ライブ タイルの外観は通知ごとに大きく変わることがありますが、名前/ロゴの場所には一貫性があります。このことにより、その情報が各タイルの同じ場所に表示されるため、ユーザーはすばやく見るだけでお気に入りのアプリを探すことができます。提供さ

れたブランド要素 (名前とロゴ) をアプリで利用しない場合は、ユーザーにとってすばやくアプリのタイルを特定することが難しくなる可能性があります。
次の画像は、ブランドの伝え方が不適切なテンプレートのテキスト要素と画像要素を使ったタイルを示しています。どちらの場合も、タイルは名前またはロゴをデザインされたまま使っています。つまり、さらにブランド化しても余分な情報となります





- オプションの "短い名前" で指定されたスペースにアプリの名前が収まらない場合は、短縮バージョンまたは意味のある頭文字を使います。たとえば、いつでも使いたくなる "Contoso Fun Game Version 3" の代わりに "Contoso Game" を使うことができます。最大ピクセル数を超える名前は途中で切り捨てられ、省略記号が付けられます。英語の場合、名前の最大長は 2 行で約 40 文字ですが、これは名前に含まれる文字によって異なります。デザインの観点から、短いアプリ名をお勧めします。また、マニフェストでさらに長いアプリの名前 ("表示名") を指定することもできます。この名前は、アプリ ビューとツールチップで使われますが、タイルでは使われません。
- タイルを広告に使わないでください。
- タイルで派手な色を使いすぎないようにします。騒ぎ立てる子供のように、注意を引くために派手に飾り立てたタイルよりも、シンプルで見やすい洗練されたデザインのタイルの方が成功します。
- テキスト コンテンツでテキストと画像を併用しないでください。テキスト コンテンツには、テキスト フィールドが含まれたテンプレートを使います。画像内のテキストは、レンダリングされたタイルのテキストほど鮮明には見えません。現在の表示に適した画像アセットが指定されていない場合は、画像は拡大される可能性があり、さらに読みにくくなることがあります。
- 緊急のリアルタイム情報をユーザーに送るときは、タイルに頼らないでください。たとえば、通信アプリでユーザーに着信を伝える場合、タイルは適切なサーフェスではありません。リアルタイム性のあるメッセージに適したメディアはトースト通知です。

- ハイパーリンク、ボタン、その他のコントロールのように見える画像コンテンツを含めないようにします。タイルではこれらの要素をサポートしておらず、タイル全体がシングルクリックの対象となります。
- 相対的なタイム スタンプや日時 (たとえば "2 時間前" など) は、時間が経過しても静的で、メッセージを不正確にするため、タイル通知で使わないでください。"11:00 A.M." などの絶対的な日時を使います。
- アプリのタイルからはアプリをホーム画面でしか起動できないため、タイルの更新は、そのホーム画面から簡単にアクセスできるアプリの要素に関するものにしてください。たとえば、ニュース アプリのタイルには、ユーザーがタイルをクリックしてアプリのホーム ページで簡単に見つけることができる記事のみを表示する必要があります。

タイル通知の更新

タイルを更新するための正しい通知方法の選択

ライブ タイルの更新には、以下の複数のメカニズムを利用できます。

- ローカル API 呼び出し
- ローカル コンテンツを使った、1 回限りのスケジュールされた通知
- クラウド サーバーから送信されるプッシュ通知
- 一定の間隔でクラウド サーバーから情報を取得する定期的な通知

使うメカニズムの選択は、表示するコンテンツとコンテンツを更新する頻度によって大きく左右されます。大部分のアプリは、起動時またはアプリ内での状態の変更時に、ローカル API 呼び出しを使ってタイルを更新すると考えられます。このようにすると、アプリの起動時と終了時に、タイルが必ず最新の状態になります。ローカル通知、プッシュ通知、スケジュールされた通知、またはポーリング通知のどれを使い、さらに単独または組み合わせて使うかは、アプリによってまったく異なります。たとえば、ゲームではプレーヤーが新しいハイ スコアに達したときに、ローカル API 呼び出しを使ってタイルを更新できます。同時に、同じゲーム アプリで、プッシュ通知を使って友人が獲得した新しいハイ スコアをプレイヤーに送信できます。

タイルの更新頻度

ライブ タイルを使う場合は、タイルの更新頻度を検討します。

- メッセージ数やゲームのプレイ順など各ユーザーに合わせたコンテンツでは、特にユーザーがタイルのコンテンツの遅れ、不正確さや不足に気付く可能性がある場合は、情報が利用可能となった時点でタイルを更新することをお勧めします。
- 天気予報の更新など、各ユーザーに合わせていないコンテンツでは、30 分に 1 回以下の頻度でタイルを更新することをお勧めします。このようにすると、ユーザーに負担をかけずにタイルが最新の状態になっていることを感じることができます。

タイル通知とバッジ通知の有効期限

タイルのコンテンツは、意味がなくなっても保持されることがないようにする必要があります。すべてのタイル通知とバッジ通知には、アプリにとって適切な有効期限を設定します。既定では、ローカルとスケジュールのタイルとバッジには期限がなく、プッシュ通知または定期的な通知を通じて送られたタイルとバッジのコンテンツの期限は送信後 3 日です。通知の有効期限が切れると、タイルまたはキューからコンテンツが削除され、ユーザーに表示されなくなります。

通知のコンテンツの期限として、特定の日時を設定できます。明示的な有効期限は、コンテンツの存続期間が決まっている場合に特に役立ちます。また、クラウド サービスが通知の送信を停止したり、アプリが長い間実行されなかったり、ユーザーがネットワークに長期間接続しなかったりする場合、明示的に有効期限を設定しておくと、システムの接続状態に関係なく、古いコンテンツが確実に削除されます。

たとえば、株式市場の取引が活発な日は、株価の更新の有効期限を送信間隔の有効期限の 2 倍に設定することをお勧めします (30 分ごとに通知を送っている場合は有効期限を通知の送信後 1 時間にするなど)。また、ニュース アプリの場合、毎日のニュースを表示するタイルの更新の有効期限は 1 日が適しています。

有効期限をどのように設定するかは、配信方法によって異なります。プッシュ通知と定期的な通知では、通知を配信するクラウド サービスとの通信に使われる HTTP ヘッダーで設定

します。ローカル通知とスケジュールされた通知では、API 呼び出しの一部として設定できます。

ロック画面でのタイルとバッジの表示

- 良いロック画面の表示の特徴
- ロック画面要求 API を使うケース

アプリがロック画面での表示に適しているかどうかを判断するには、ロック画面の操作と制限を理解する必要があります。ここではロック画面の要約を示します。詳しくは、「ロック画面の概要」をご覧ください。

- 最大 7 つのアプリ バッジをロック画面に表示できます。バッジの情報には、アプリのスタート画面のタイルに表示されるバッジの情報が反映されます。バッジ (グリフまたは数値のどちらか) には、バッジが関連付けられたアプリを特定できるモノクロのアイコン (ロゴ イメージ) が付随します。
- これら 7 つのアプリのうち 1 つだけが詳しい状態スロットを占有し、アプリの最新のタイルの更新をテキスト コンテンツで表示できます。
- ロック画面の詳しい状態タイルには、タイルの更新に含まれた画像は表示されません。
- ユーザーは、ロック画面に情報を表示できるアプリと、詳しい状態を表示できるアプリを選びます。
- ロック画面に表示されるすべてのアプリは、バックグラウンド タスクを実行することもできます。バックグラウンド タスクを実行できるすべてのアプリは、ロック画面に表示されます。アプリは、同時にロック画面のスロットを要求しないでバックグラウンド タスクを使うことはできません。
- 通知キューは、ロック画面の詳しい状態タイルではサポートされません。最新の更新のみが表示されます。
- ロック画面に表示されるアプリでは、そのアプリのマニフェストで **[トースト対応]** オプションを "はい" に設定している限り、ロック画面の表示中、受け取ったトースト通知がロック画面に表示されます。ロック画面に表示されるトーストは、別の場所で表示されるトーストと同じです。

- タイルの更新、バッジの更新とトースト通知は、ロック画面専用に設計されていたり、またはロック画面だけに送られるわけではありません。送信者には、デバイスが現在ロックされているかどうかはわかりません。ロック画面に表示されるアプリの場合、通知はスタート画面とロック画面の両方に反映されます。

良いロック画面の表示の特徴

アプリをロック画面に表示するには、必ずユーザーが明示的な許可を与える必要があります。それには、ユーザーがアプリからの要求に応えるか (ユーザーに要求できるのは 1 回限り)、**[設定]** で手動で許可します。アクセス許可を与えることで、ユーザーはアプリから送られる情報が自分にとって重要であることを示し、アプリはそれに応える必要があります。そのため、アプリ自体がロック画面での表示に適しているかどうかを検討する必要があります。

ロック画面での表示に適した候補には、次のような特性があります。

- 情報がすぐに理解できる
- 情報は常に最新にする
- 追加のコンテキストがなくても、情報が理解できる
- ユーザーに合わせた便利な情報
- 役立つ情報に絞った的確な表示
- トースト通知でのみ到着時にサウンドを再生する

情報がすぐに理解できる

ロック画面が表示されると、ユーザーが現在デバイスを操作していないことを表します。そのため、アプリがロック画面に表示する更新情報は、ユーザーの注意を引き、ひとめで理解できるものにする必要があります。似たような例として、携帯電話の着信を考えてみます。電話をひとめ見れば発信者がわかり、応答するか、そのままボイス メールに送ることができます。ロック画面に表示される情報も、携帯電話の表示と同じように簡単に理解でき、処理されるものにしてください。他のすべての特徴を使って、これを実現できます。

情報は常に最新にする

良いバッジの更新、タイルの更新、トースト通知は、スタート画面またはロック画面アプリのどちらに表示されても、すべてアクション可能です。このような通知が提供する情報に基づいて、ユーザーは新しいメールを読む、ソーシャル メディアの投稿にコメントするなど、アプリを起動して応答するかどうかを判断できます。ロック画面では、デバイスのロックの解除も必要です。そのため、ユーザーが十分な情報に基づいて判断できるよう、情報は最新にする必要があります。ユーザーは、ロック画面に表示されるアプリの情報が最新でないことに気付くようになると、アプリを信頼しなくなり、より信頼性の高い役立つアプリを探してロック画面のスロットに配置するようになるでしょう。

良い例: 最新の情報

- メッセージング アプリは、新しいメッセージが届いたときに通知を送ります。その通知が無視されると、アプリは未読のメッセージ数でバッジを更新します。ユーザーが操作中である場合、画面をオンにしてメッセージの重要度を判断し、すぐに応答するか、後で応答するかを選ぶことができます。ユーザーが操作中でない場合は、戻ったときに正確な未読メッセージの数を確かめます。
- メール アプリはバッジを使って未読メールの数を表示します。 新しいメールが到着すると、バッジがすぐに更新されます。ユーザーはすぐに画面をオンにして未読メールの数を確かめることができます。また、その数が正確であることが保証されます。ユーザーには、デバイスのロックを解除してメールを読むかどうかを判断する情報があります。

悪い例: 古い情報

- メッセージング アプリは、30 分に 1 回だけ未読のメッセージ数でバッジを更新します。ユーザーは、デバイスのロックを解除するかどうかを判断する際に、バッジの数値は参考になりません。
- 詳しい状態スロットを使う天気予報アプリが、重大な天気予報の警報が期限切れになった後も、その警報を表示し続けます。これはユーザーに誤った情報を与えるだけでなく、テキストに警報の期限が示されていると、古い情報であることがユーザーにはっきりわかるため、非常に良くない例です。アプリで常に適切な情報を知ることができるというユーザーからの信頼が失われます。情報が期限切れになったとき、アプリはその情報をクリアしておく必要があります。

- カレンダー アプリが、期日を過ぎた予定を表示し続けています。この場合も、情報が期限切れになったときに、アプリはその情報をクリアしておく必要があります。

追加のコンテキストがなくても、情報が理解できる

ロック画面には、次のようなコンテキスト情報は表示されません。

- アプリに詳しい状態の表示が許可されていない場合の、バッジに対応するタイル。詳しい状態が表示される場合でも、バッジはタイルからは物理的に切り離されています。バッジの横のロゴ イメージが、そのアプリを示す唯一の要素です。
- タイル更新時の画像。更新のテキスト部分だけが、詳しい状態スロットに表示されます。
- 通知キュー。最新の更新だけが、詳しい状態スロットに表示されます。

そのため、更新は、利用できる追加のコンテキストがスタート画面になくても、ユーザーが理解できるようにする必要があります。繰り返しますが、通知をロック画面専用にすることはできません。したがって、アプリによる更新のやり取りは、必ず "追加のコンテキストがなくても理解できる" というルールに従う必要があります。

注 詳しいタイルとは異なり、トーストでは画像 (存在する場合) とテキストの両方が表示されます。ロック画面に表示されるトーストはそれ以外の場所に表示されるトーストと同じであるため、コンテキストは失われません。

良い例: 追加のコンテキストがなくても理解できる

- メール アプリはバッジを使って未読メールの数を表示します。スタート画面のタイルには、最新のメールからのテキスト スニペットや送信者の画像など、詳しい情報が表示されることがありますが、バッジではこのような追加情報がなくても理解できる情報を伝えます。
- ソーシャル ネットワーキング アプリが、詳しい状態スロットを使って、ユーザーに友人の最近の活動を知らせます。友人がユーザーにメッセージを送ると、通知のテキストに友人の名前が含まれます (たとえば、"カイルから新しいメッセージが届きました!" など)。スタート画面では、通知に表示される友人の画像によって、ユーザーに優れたエクスペリエンスを提供できます。ロック画面では、画像は表示さ

れませんが、だれがメッセージを送信したかがテキストからはっきりとわかります。

悪い例: 追加のコンテキストがないと理解できない

- メッセージング アプリが最新の受信メッセージでタイルを更新し、送信者の画像とメッセージのテキストだけを表示します。スタート画面では、ユーザーは、だれがメッセージの送信者であるかはっきりわかります。ロック画面では、送信者の画像がないため、ユーザーはだれがメッセージを送信したのかわかりません。
- ソーシャル ネットワーキング アプリがフォト コラージュでタイルを更新します。テキストはありません。スタート画面では、これは楽しく、生き生きとしたタイルです。ロック画面では、タイルの更新にテキストが含まれていないため、何も表示されません。

ユーザーに合わせた便利な情報

ロック画面の主な目的のうち 2 つは、ユーザーにカスタマイズされたサーフェスを提供することと、アプリの更新プログラムを表示することです。アプリがロック画面での表示に適しているかどうかを判断する際に、これらの目的を両方とも検討してください。

ロック画面に表示されるアプリは非常に特殊です。同時に 7 個だけロック画面に表示できます。貴重なロック画面のスロットの 1 つを割り当てることにより、ユーザーは、頻繁にデバイスを使っていない場合でも、そのアプリから送られる情報が重要で確認する必要があることを示しています。そのため、アプリが提供する情報は、ユーザーに合わせると同時に便利である必要があります。

注 デバイスがロックされるとロック画面が表示されるというのが決まりです。ロック画面のコンテンツを表示するうえで、ログインなどのセキュリティ上の操作は必要ありません。したがって、ロック画面に表示される情報は個人向けにカスタマイズされるのが理想的ですが、だれでもその情報を見ることができるということに注意してください。

良い例: ユーザーに合わせた情報

- メール アプリは、ユーザーのアカウントにある未読メールの数を表示します。

- メッセージング アプリは、ユーザーに送信された未読メールの数を表示します。
- ニュース アプリは、ユーザーがお気に入りとしてフラグを付けたカテゴリにあるニュース記事の数を表示します。

悪い例: ユーザーに合わせていない情報

- ニュース アプリは、ユーザーが示した好みをまったく考慮しないで、ニュース サービスから送られるすべての新しい記事の合計数を表示します。
- ショッピング アプリは、セールの通知を送りますが、ユーザーが示した好みの商品やカテゴリに基づいていません。

役立つ情報に絞った的確な表示

情報は、変更が発生した場合だけ表示する

前に説明したように、目的は、ロック画面で情報がひとめで理解されるようにすることです。そのために、アプリが現在バッジを表示していない場合は、ロック画面上でバッジが本来表示される場所にすき間が残されます。このことにより、ユーザーは注意が必要な情報に気付きやすくなります。イベント後のバッジとロゴの外観は、新しい情報を伝えることなく最初からその場所に表示されていた場合よりも目立ちます。

状態を示すという目的のためだけに、状態を表示しないでください。長期間続く状態や変更されることのない状態は、ロック画面を煩雑にするだけで、より重要な情報を目立たなくします。ユーザーが気付くべきイベントが発生したときだけ、バッジを表示します。タイルの更新も同じです。古くなった通知コンテンツはタイルから削除します。削除すると、スタート画面ではタイルは既定の画像に戻り、ロック画面では何も表示されません。

良い例: 有益な場合にのみ表示される情報

- メール アプリは、未読メールがあるときにのみバッジを表示します。新しいメールが到着すると、バッジは更新されて表示されます。
- メッセージング アプリは、ユーザーがメッセージを受信できないときだけ接続状態を表示します。"接続された" 状態は、アプリの既定の状態であるという前提のため、その情報を伝える意味はありません。"すべて順調" という通知は、アクション

可能な通知とはいえません。ただし、メッセージを受信できないときにユーザーに通知することは有益で、アクション可能な情報といえます。

悪い例: 長期間続く状態

- メール アプリまたはメッセージング アプリで、未読メールの数が表示されないと、新しいメールまたはメッセージが到着するまで接続状態を表示します。このようにすると、バッジが常に表示されているため、ユーザーは新しいメッセージがあるかどうかをひとめで確認しにくくなります。
- カレンダー アプリが、予定がないことを示すメッセージを表示します。この場合も、常に何か情報が詳しい状態スロットに表示されるため、ひとめ見て確認できるという詳しい状態スロットの操作性が低下します。

トースト通知でのみ到着時にサウンドを再生する

バッジまたはタイルが更新されたときにサウンドを再生するコードをアプリに埋め込まないでください。トーストについては、到着時にサウンドを再生するように設計できます。

この記事で説明しているガイダンスに従うと、ロック画面に正しい情報を正しい方法で表示するアプリを作成できるようになり、アプリに対するユーザーの満足度と信頼を高めることができます。

ロック画面要求 API を使うケース

アプリが正常に機能するためにバックグラウンド権限が必要な場合にのみ、ロック画面要求 API (**RequestAccessAsync**) を呼び出してください。利用できるバックグラウンド スロットは 7 つだけであるため、ユーザーは、正常に機能するにはバックグラウンド権限が必須であるアプリと、(バックグラウンド権限があれば機能が追加される場合でも) バックグラウンド権限がなくても正常に機能するアプリを見分ける必要があります。

アプリがユーザーの要求を満たすうえでバックグラウンド権限が不可欠な場合にのみ、要求 API を使って、アプリをロック画面に配置するようユーザーに求めることをお勧めします。

バックグラウンド権限がなくてもユーザーの要求を満たすことができる場合は、アプリをロック画面に配置するようユーザーに対して明示的に求めないようにしてください。代わり

に、ユーザーが **[PC 設定]** の **[パーソナル設定]** ページを通じてアプリをロック画面に配置できるようにします。

要求 API を呼び出す必要のあるアプリの例:

- アプリがフォアグラウンドに存在しないときにメッセージを受け取るためにバックグラウンド権限が必要なメッセージング アプリ
- アプリがフォアグラウンドに存在しないときにユーザーの受信トレイを同期するためにバックグラウンド権限が必要なメール アプリ

要求 API を呼び出さないようにするアプリの例:

- 予報を更新するためにバックグラウンド アクティビティではなく定期的な通知を使う天気予報アプリ
- バッジで示される新しい記事の数を特定の時刻に更新するニュース アプリ

注 アプリには、ロック画面にそのアプリを追加するようユーザーに求めるダイアログを実装できません。アプリでロック画面へのアクセスが正常に動作する必要がある場合は、ロック画面要求 API によって表示されるダイアログを使う必要があります。以前にユーザーがこのダイアログでアプリに対するロック画面の権限を拒否した場合は、このダイアログが再表示されないことがあります。この場合は、アプリのインライン テキストを使って、**[PC 設定]** の **[パーソナル設定]** ページにユーザーを誘導し、手動でアプリをロック画面に追加してもらうこともできます。

セカンダリ タイルのガイドライン

セカンダリ タイルを使うと、スタート画面から Windows ストア アプリ内の特定の領域に、一貫した方法で効率的に直接アクセスすることができます。ユーザーはセカンダリ タイルをスタート画面に"ピン留め"するかどうかを選択できますが、アプリ内のピン留めできる領域は開発者によって決められます。詳しくは、「セカンダリ タイルの概要」をご覧ください。Windows ストア アプリでセカンダリ タイルを有効にし、関連する UI を設計するときは、次のガイドラインを考慮してください。

注 セカンダリ タイルをスタート画面にピン留めできるのはユーザーだけです。アプリでプログラムによってピン留めすることはできません。タイルの削除もユーザーがコントロールし、セカンダリ タイルをスタート画面や親アプリ内から削除することができます。

推奨と非推奨

アプリでセカンダリ タイルを有効にするときは、以下の推奨事項を考慮してください。

- 対象のコンテンツがピン留めできる場合は、セカンダリ タイルを作る "[スタート画面にピン留めする]" ボタンをアプリ バーに表示する必要があります。
- ユーザーが "[スタート画面にピン留めする]" ボタンをクリックしたときに表示されるフライアウトを作成します。このフライアウトでは、セカンダリ タイルをスタート画面に追加してもよいかユーザーに確認します。例として、ESPN アプリの確認フライアウトを次に示します。

- 対象のコンテンツが既にピン留めされている場合は、アプリ バーの "[スタート画面にピン留めする]" ボタンを "[スタート画面からピン留めを外す]" ボタンに置き換えます。"[スタート画面からピン留めを外す]" ボタンは、既にあるセカンダリ タイルを削除します (ユーザーの確認を得た後)。
- 対象のコンテンツがピン留めできない場合は、"[スタート画面にピン留めする]" ボタンを表示しません (または、"[スタート画面にピン留めする]" ボタンを無効にします)。
- "[スタート画面にピン留めする]" ボタンと "[スタート画面からピン留めを外す]" ボタンには、システムが提供するグリフを使います (**Windows.UI.Xaml.Controls.Symbol** または **WinJS.UI.AppBarIcon** のピン留めとピン止め解除のメンバーをご覧ください)。
- ボタンのテキストは標準の "スタート画面にピン留めする" と "スタート画面からピン留めを外す" を使います。システムによって提供されるピン留めとピン留め解除のグリフを使うときは、既定のテキストをオーバーライドする必要があります。
- "次のトラックにスキップ" タイルのように、親アプリと対話するための、事実上のコマンド ボタンとしてセカンダリ タイルを使わないでください。

その他の使い方のガイダンス

開発者向け

- アプリの起動時には、常にセカンダリ タイルを列挙する必要があります。セカンダリ タイルの追加や削除を把握していない場合があるためです。スタート画面のアプリ バーを使ってセカンダリ タイルを削除すると、タイトルも削除されます。セカンダリ タイルによって使われていたリソースは、アプリ自体で解放する必要があります。クラウドを通じてセカンダリ タイルがコピーされた場合、セカンダリ タイル上のタイルまたはバッジの現在の通知、スケジュールされた通知、プッシュ通知チャネル、定期的な通知で使われる URI (Uniform Resource Identifier) はタイルと共にコピーされないので、再設定する必要があります。
- セカンダリ タイルの作成時には **RequestCreateForSelectionAsync** クラスを使います。このクラスを使うと、確認ポップアップを想定どおりに配置できるため、ユーザー エクスペリエンスが向上します。

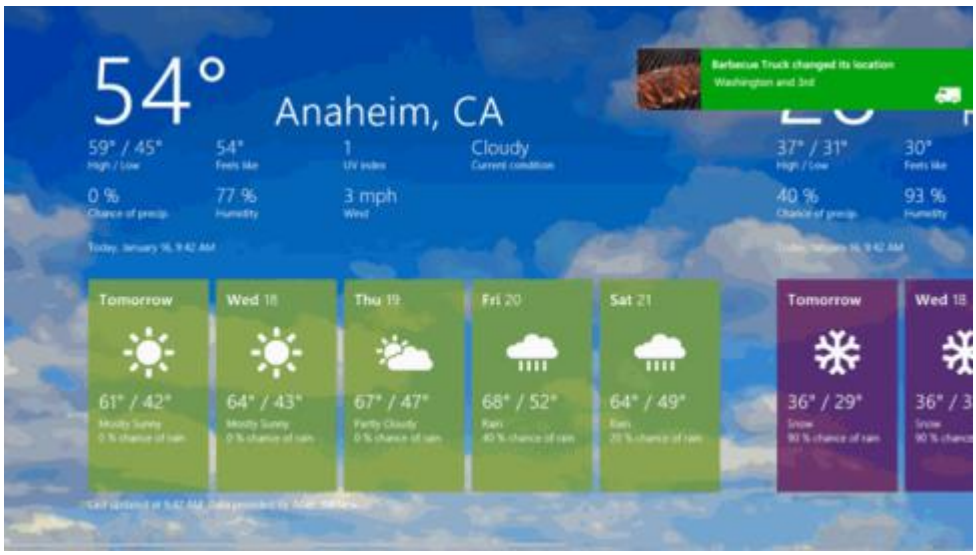
- アプリを公開した後でタイルの既定の画像アセットの名前を変更しないでください。セカンダリ タイルが最初の通知を受け取るまでと、表示する通知がないときは、既定の画像が表示されます。想定される画像が見つからない場合は、空のタイルが表示されます。
- アプリでは、セカンダリ タイルに意味のある再作成可能な一意の ID を使う必要があります。この点が重要な理由は次のとおりです。
    - アプリを別のコンピューターにインストールしたときに、ユーザーはセカンダリ タイルを再取得できます。アプリにとって意味のある予測可能なセカンダリ タイル ID を使うことにより、新しいコンピューターの新規インストールでセカンダリ タイルが表示されたときに、アプリはそれらのタイルで実行する処理を判断できるようになります。
    - 実行時に、アプリは特定のタイルが存在するかどうかを照会できます。
    - セカンダリ タイルのプラットフォームに、特定のアプリに属するすべてのセカンダリ タイルのセットを返すよう要求できます。これらのタイルに意味のある一意の ID を使うことにより、アプリはセカンダリ タイルのセットを調べ、適切なアクションを実行できます。たとえば、ソーシャル メディア アプリの場合、ID によってタイルの作成対象である個々の連絡先を識別できます。
- スタート画面のすべてのタイルと同様に、セカンダリ タイルは新しいコンテンツで頻繁に更新できる動的な表示機能です。セカンダリ タイルでは、他のタイルと同じメカニズムを使って通知や最新情報を表示できます。アプリを実行していないときにタイルを更新するには、セカンダリ タイルで Windows プッシュ通知サービス (WNS) を使ってチャネルの URI を要求し、URI を開く必要があります。詳しくは、「通知チャネルを要求、作成、保存する方法」を参照してください。

トースト通知のガイドライン

このトピックでは、トースト通知の用途を説明し、トースト通知の作成方法と送信方法について推奨事項を示します。

例

お気に入りの移動式屋台が出店場所を変えたときにトースト通知を受け取るようにしています。



アプリにトースト通知を含めるかどうか

トースト通知を使うと、即時性が必要な事項や個人的に関係のある事項について、別のアプリを使っている場合や、スタート画面、ロック画面、デスクトップが表示されている場合でも関係なく、ユーザーに対して通知できます。たとえば、トースト通知を使ってユーザーに次のようなことを通知できます。

- VOIP の着信呼び出し
- 新着インスタント メッセージ
- 新着テキスト メッセージ
- カレンダーの予定やその他のアラーム
- その他のユーザーが要求した個人的に重要な通知

トースト通知を受け取るには、ユーザーがオプトインする必要があり、また、ユーザーはいつでもトースト通知を無効にできることを忘れないでください。

推奨と非推奨

アプリにトースト通知を追加するときは、以下の推奨事項を考慮してください。

- ユーザーがトースト通知をクリックしたときに、アプリの適切な移動先に移動します。通知は厳密な情報更新ではなく、コンテキストの切り替えをユーザーに促すものと考えてください。
- 情報が重要な場合は、トースト通知で得られる情報をユーザーが入手できる代替方法を用意します。たとえば、アプリのライブ タイルやアプリ内に関連情報を表示することができます。
- 短期間に複数の関連する更新が発生した場合は、まとめて 1 つのトースト通知にします。たとえば、3 つの新しい更新が同時に到着した場合、アプリまたはアプリ サーバーは、3 つの個別の通知ではなく、3 つの新しい更新があることを示す 1 つの通知を表示する必要があります。
- 情報はできるだけシンプルな形で提供します。コンテンツでヘッドラインが不要な場合は、ヘッドラインを省略します。"ダウンロードが完了しました" などのメッセージはこれで完結しているので、追加の表示は不要です。
- 画像はメッセージに明確な付加価値を付ける場合に使います (メッセージの送信者の写真など)。
- 有効でなくなった通知は表示しないようにします。たとえば、相手が電話を切ったり、ユーザーが別のデバイスで既に通話している場合は、着信呼び出しを非表示にします。アプリが実行中のときしか通知を非表示にできないことに注意してください。
- 重大な情報をユーザーに通知するためにトースト通知を使わないでください。代わりに、重大な通知が確実に目にとまるように、フライアウト、ダイアログ、アプリバー、その他のインライン要素を使ってアプリ内でユーザーに通知します。
- "... するためにここをクリックする" ことをユーザーに伝えるテキストを含めないでください。すべてのトースト通知は、クリックまたはタップ操作によって関連アプリに移動することを前提としています。

- トースト通知を使って、一時的な障害やネットワーク イベント (接続の切断など) をユーザーに通知しないでください。
- 株価情報など、大量の通知を伴うものにトースト通知を使わないでください。
- トースト通知を使って、定期メンテナンス イベント (ウイルス スキャンの完了など) をユーザーに通知しないでください。
- アプリがフォアグラウンドで動作していて、コンテキストにより適したサーフェス (インライン要素、フライアウト、ダイアログ、アプリ バーなど) を使うことができる場合は、トースト通知を表示しないでください。たとえば、ビュー内で進行中の会話に関連する追加のインスタント メッセージは、個々の新しいメッセージが含まれたトースト通知を継続して表示するのではなく、会話をインラインで更新する必要があります。アプリケーションが実行中のとき、**PushNotificationReceived** イベントをリッスンして、プッシュ通知を中断します。
- 通知の画像フィールドにアイコンやアプリのロゴなどの汎用画像を追加しないでください。
- 通知のテキストにアプリ名を含めないでください。ユーザーはアプリのロゴでアプリを識別します。アプリのロゴはトースト通知に自動的に表示されます。
- ユーザーがトースト通知を無効にすることを選んでいる場合は、アプリを使って、トースト通知を有効にすることをユーザーに要求しないでください。アプリはトースト通知なしで動作することを求められています。
- バルーン通知のシナリオをトーストに自動的に移行しないでください。ユーザーが全画面表示のエクスペリエンス (デスクトップ スタイル アプリのみ) に没入していないときには、バルーン通知の方が適している場合もあることを考慮します。
- その日の概況など、リアルタイムでない情報にトースト通知を使わないでください。
- どうしても必要な場合以外は、トースト通知を非表示にしないでください。
- ユーザーが通知を求めていないものを通知しないでください。たとえば、知り合いがオンラインになるたびに通知されることをすべてのユーザーが望んでいるわけではありません。

ファイル、データ、グローバリゼーションのガイドライン

世界市場に向けて容易に提供できるアプリの設計方法について検討します。アプリのリソース (文字列や画像など) をコードから分離します。これにより、ローカライズが容易になります。

このセクションの内容

トピック	説明
アプリ リソース	このトピックでは、Windows ストア アプリでアプリ リソースを使うためのベスト プラクティスについて説明します。
グローバリゼーションとローカリゼーション	広範なユーザー向けにアプリをグローバル化したり、特定の市場を対象にアプリをローカライズするときは、次のベスト プラクティスに従ってください。

アプリ リソースのガイドライン

文字列や画像などのアプリ リソースをコードから分離することは、アプリの保守やローカライズのプロセスを簡単にする方法の 1 つです。このトピックでは、Windows ストア アプリでアプリ リソースを使うためのベスト プラクティスについて説明します。

アプリ リソースの詳細と、リソース管理 API の使い方を説明したトピックについては、次をご覧ください。

- アプリ リソースの定義 (JavaScript と HTML を使った Windows ストア アプリ)
- アプリ リソースの定義 (C#/VB/C++ と XAML を使った Windows ストア アプリ)

推奨と非推奨

リソースの作成

- UI 文字列や画像などのリソースをコードに格納しないでください。代わりに、.resjson ファイルや .resw ファイルなどのリソース ファイルに格納してください。
- 修飾子を使って、さまざまな表示スケール、UI 言語、ハイ コントラスト設定に合わせたファイルと文字列リソースをサポートします。
- アプリ マニフェスト (package.appxmanifest) に既定の言語を設定します。
- 文字列リソースには、既定の言語であっても、その言語タグの名前を持つファイルまたはフォルダーが必要です。
- ローカライズ担当者に向けたコメントを文字列リソースに追加します。

リソースの名前付けについて詳しくは、「修飾子を使ってリソースに名前を付ける方法 (HTML)」または「修飾子を使ってリソースに名前を付ける方法 (XAML)」をご覧ください。

リソースの参照

- リソースを参照するための一意のリソース識別子をコードとマークアップに追加します。

- マークアップ、コード、またはマニフェスト ファイルで修飾子を除いて画像を参照します。
- システムが変更されて、異なるセットの修飾子の使用が始まるときに発生するイベントをリッスンします。正しいリソースが読み込まれるようにドキュメントを再処理します。

UI リソースの翻訳とローカライズの準備について詳しくは、「アプリのグローバル化 (HTML)」または「アプリのグローバル化 (XAML)」をご覧ください。世界中のユーザーに適合するアプリを作成するための推奨事項については、「グローバリゼーションのガイドライン」をご覧ください。

グローバリゼーションのガイドライン

広範なユーザーや市場向けにアプリをグローバル化したり、特定のユーザーや市場を対象にアプリをローカライズするときは、次のベスト プラクティスに従ってください。

グローバリゼーション機能をアプリに追加する方法については、以下をご覧ください。

- アプリのグローバル化 (JavaScript と HTML を使った Windows ストア アプリ)
- アプリのグローバル化 (C#/VB/C++ と XAML を使った Windows ストア アプリ)

推奨と非推奨

グローバリゼーション

グローバル化に適した UI の用語と画像が選択され、**Globalization** API を使ってアプリ データがフォーマットされ、場所や言語に基づく前提のない、異なる市場に簡単に適応できるアプリを準備します。

プラクティス	説明
数値、日付、時刻、住所、電話番号には正しい形式を使う。	数値、日付、時刻などのデータに使われる形式は、カルチャ、地域、言語、市場により異なります。数値、日付、時刻などのデータを表示する場合は、**Globalization** API を使ってユーザーが好む形式を取得してください。
国際的な用紙サイズをサポートする。	最も一般的な用紙サイズは国によって異なるため、用紙サイズによって変化する機能 (印刷など) を含める場合には、必ず一般的な国際サイズをサポートし、テストしてください。
国際的な計測単位と通貨をサポートする。	使われる単位と尺度は国によって異なりますが、最も使われているのはメートル法とヤード ポンド法です。長さ、温度、範囲などの計測を扱う場合は、**Globalization** 名前空間を使って正しいシステム計測を取得してください。アプリが通貨の表示をサポートする場合は、必ず正しい書式設定を使ってください。**CurrenciesInUse** プロパティを使って、ユーザーの地理的な地域の通貨を取得することもできます。
テキストとフォントを正しく表示する。	テキストに適したフォント、フォント サイズ、方向は、市場によって異なります。
文字エンコードに Unicode を使う。	既定では、最近のバージョンの Microsoft Visual Studio は、すべてのドキュメントに Unicode 文字エンコードを使います。別のエディターを使っている場合は、適切な Unicode 文字エンコードでソース ファイルが保存されるようにしてください。Windows ランタイム API はどれも、UTF-16 エンコードの文字列を返します。
入力の言語を記録する。	アプリがユーザーにテキスト入力を求めるときに、入力の言語が記録されるようにします。こうすると、その入力が後で表示されるときに適切な書式設定でユーザーに提示されます。現在の入力言語の取得には、**CurrentInputMethodLanguage** プロパティを使います。

言語からユーザーの位置を想定する、または位置からユーザーの言語を想定することはしない。	Windows では、ユーザーの言語と位置は別の概念です。特定の地理的な言語バリアント (en-gb (英国で話される英語) など) を話しても、住んでいる国または地域はまったく異なる場合があります。UI テキストなどのためにアプリがユーザーの言語について認識する必要があるか、ライセンス問題などのためにアプリがユーザーの位置について認識する必要があるかを検討してください。
俗語と比喩を使わない。	一部のカルチャや年齢などの集団にしか伝わらない言葉は、その集団の人しか使わないので、理解や翻訳が難しい場合があります。同様に、比喩も人によって伝わったり伝わらなかったりします。たとえば、"ブルーバード" はスキーをする人には伝わりますが、スキーをしない人には伝わりません。アプリをローカライズしてくだけた表現を使うことを計画している場合は、ローカライズ担当者に翻訳対象の意味と口調を十分に説明してください。
専門的な用語、省略形、略語を使わない。	専門用語は、専門知識のないユーザーや他のカルチャまたは地域の人々には意図が伝わりにくく、翻訳も困難です。このような言葉は日常会話では使われません。専門用語は、ハードウェアとソフトウェアの問題を特定するため、エラー メッセージ内でよく使われます。専門用語が必要な場合があるかもしれませんが、普通の言葉に置き換えることが望まれます。
不快感を与えかねない画像は避ける。	自分が所属するカルチャでは妥当な画像でも、別のカルチャでは不快感を与えたり、誤って解釈されたりすることがあります。宗教的なシンボル、動物、国旗や政治運動に関連付けられる色の組み合わせなどは避けてください。
地図や、地域についての言及では、政治的侵害を避ける。	地図には論争の的になっている地域や国境が含まれている可能性があり、それらはしばしば政治的な侵害のきっかけになります。国家の選択に使う UI は、必ず "国/地域" という名称にしてくださ

	い。(住所フォームなどで) "国" という名称の一覧に領有権未決の領域を含めると、トラブルになりかねません。

言語タグを比較する目的で文字列比較を単独で使わない。	BCP-47 言語タグは複雑です。言語タグの比較では、スクリプト情報、前のタグ、複数の地域バリアントの対応付けに伴う問題など、多数の問題が発生します。Windows のリソース管理システムでは、対応付けが自動的に行われます。開発者はどの言語で作られたリソース セットでも指定でき、システムがユーザーとアプリのために適切なものを選びます。
並べ替えが常にアルファベット順で行われると想定しない。	ラテン文字を使わない言語の場合、並べ替えは発音、ペン ストロークの数などの要素に基づいて行われます。ラテン文字を使う言語でも、常にアルファベット順の並べ替えを行うわけではありません。たとえば、一部のカルチャでは電話帳はアルファベット順では並んでいない場合があります。システムによって並べ替えが自動的に行われますが、自分で独自の並べ替えアルゴリズムを作る場合は、必ず、アプリの対象市場で使われている並べ替え方法を考慮してください。

ローカライズ

プラクティス	説明
UI 文字列や画像などのリソースをコードから分離する。	アプリは、文字列や画像などのリソースがコードから分離されるように設計してください。こうすることで、さまざまなスケール ファクター、アクセシビリティ オプション、ユーザーとコンピューターに関する多くのコンテキストに対して、それらの保守、ローカライズ、カスタマイズを個別に行うことができます。 文字列リソースはアプリのコードから分離し、言語に依存しない単一のコードベースを作ってください。常にアプリ コードとマークアップから文字列を分離し、リソース ファイル (ResW や ResJSON ファイル) に入れてください。 Windows のリソース インフラストラクチャを使って、ユーザーの実行時環境に最適なリソースが選ばれるように処理してください。詳しくは、「アプリ リソースのガイドライン」をご覧ください
他のローカライズ可能なリソース ファイルを分離する。	ローカライズが必要な他のファイル (翻訳するテキストを含んだ画像やカルチャ上の配慮のために変更が必要な画像など) は、言語名でタグ化されたフォルダーに入れてください。
既定の言語を設定し、既定の言語で作られているものも含め、すべてのリソースをマークする。	アプリの既定の言語は常に適切に設定してください。アプリでサポートされる言語のどれもユーザーが話さない場合、使われる言語は既定の言語によって決定されます。既定の言語リソースをその言語でマークしてください (例: en-us/Logo.png)。こうすることで、システムはリソースで使われている言語と、個々の状況でそのリソースがどのように使われるかを判断できます。

ローカライズが必要なアプリ リソースを特定する。	他の市場向けにアプリをローカライズすることになった場合には、何を変更する必要があるでしょうか。テキスト文字列を他の言語に翻訳する必要があります。他のカルチャに合わせて画像を変更する必要がある場合もあります。アプリが使う他のリソース (オーディオやビデオなど) にローカライズがどのような影響を与えるかを考慮してください。
リソースを参照するには、コードとマークアップでリソース識別子を使ってください。	文字列リテラル、または画像の特定のファイル名をマークアップに含めるのではなく、リソースの参照を利用してください。XAML については [link] を、HTML については [link] をご覧ください。必ず、リソースごとに一意の識別子を使ってください。
テキスト サイズを拡大できるようにする。	翻訳されるとテキスト サイズが大きくなる可能性があるため、テキスト バッファーは動的に割り当ててください。静的なバッファーを使う必要がある場合は、(英語文字列の長さを倍にするなどの方法で) それらを特大サイズにし、文字列が翻訳された時点で起きる可能性がある拡大に対応してください。 ユーザー インターフェイスに利用可能な領域が制限されることもあります。ローカライズされた言語に対応するには、文字列の長さが、日本語に必要となりそうな長さよりも約 40% 長くします。単一の語句のように非常に短い文字列の場合、必要領域が約 300% も増える可能性があります。さらに、コントロール内で複数行のサポートとテキストの折り返しを有効にすると、各文字列を表示するための領域を増やすことができます。

左右反転をサポートする。	テキストの配置と読み取りは、英語のように左から右の順にも、アラビア語やヘブライ語のように右から左の順 (RTL) にも行うことができます。読み取り順が自国語とは異なる言語に製品をローカライズする場合は、UI 要素のレイアウトが左右反転をサポートする必要があります。戻るボタン、UI 切り替え効果、画像などのアイテムですら、左右反転が必要になることがあります。
文字列にコメントする。	文字列に適切にコメントを入れ、翻訳が必要な文字列だけをローカライズ担当者に提供してください。過剰なローカライズは、よく問題を引き起こします。
短い文字列を使う。	文字列を短くすると翻訳が簡単になり、翻訳データを再使用できます。翻訳データを再使用すると、同じ文字列はローカライズ担当者に再び送られることがないため、コストを節約できます。8,192 文字を超える文字列は、一部のローカライズ ツールではサポートされない可能性があります。このため、文字列は 4,000 文字以内に抑えてください。
文全体が入った文字列を提供する。	単語の訳は文におけるその位置によって変化する可能性があるため、文を個々の単語に分割せず、文全体が入った文字列を提供してください。1 つのフレーズが複数のパラメーターから構成される場合、どの言語でもパラメーターの順序は変わらないと想定してはなりません。
画像ファイルとオーディオ ファイルをローカライズ用に最適化する。	画像内にテキストを入れることやオーディオ ファイルに音声を入れることを避けると、ローカライズ コストが抑えられます。読み取り順が自国語とは異なる言語にローカライズする場合は、左右対称の画像や効果を使うと左右反転をサポートしやすくなります。
文字列は異なるコンテキストで再使用しない。	文字列は異なるコンテキストで再使用してはなりません。"オン" や "オフ" などの簡単な語句でも、コンテキストに基づいて別の翻訳がなされる可能性があります。