Enable New Features for an Amazing User Experience!

Redesign Your iPad* App for Windows* 8

Jorge A. Toro, Ph.D.
Integrated Computer Solutions

www.ics.com
Introduction/Background

Recent advances in technology have sparked the introduction of innovative devices with new and enhanced experiences: watches, digital cameras, mp3 players, remote controls, e-book readers, smartphones, and GPS devices to name a few. Many are now equipped with features and capabilities that were once considered possible only in sci-fi movies. Most of these technological advances are leveraging touch as a key element to create more immersive and responsive experiences for their users. Personal computing has also matured; in fact, markets today are flooded with ultra-portables, convertibles, Ultrabook™ devices, rugged laptops, tablets, and smartphones, many of which use touch as the primary channel of interaction.

The iPad® is a line of tablet computers created by Apple Inc., conceived as a device that is easy to carry, learn and use yet powerful enough to create fun, engaging experiences. Its user interface is based on the concept of direct manipulation via multi-touch gestures. iPads run on Apple's iOS® operating system, which is also used on iPhone®, iPod®, and Apple TV devices. It is derived from OS X®, the current operating system used on Apple computers.

Technology advances coupled with the evolution of touch providing a new way to interact with modern devices seem to indicate that users are also evolving—transitioning away from a static Desktop environment to a more flexible, lighter, mobile one.

Microsoft Windows® is the most popular desktop and laptop operating system in the world with more than a 90% presence in the desktop market, a market that has been dominated by keyboard and mouse interactions. It is no surprise, however, that the increasing popularity of touch-based experiences has definitely influenced user preferences, behaviors, and expectations toward computing devices. In fact, a recent study¹ conducted by Intel revealed that 80% of PC users prefer touchscreens to traditional keyboard mouse interactions. Naturally, the need for a revised, improved operating system that leverages and optimizes the touch experience for PC users is clear. The latest release of Windows 8, optimized for touch experiences, has opened new possibilities in the mobile ecosystem as an alternative to Android*- and iOS-based devices.

The following case study is intended to help you understand the anatomy of a Windows Store app as well as the design challenges involved in the migration of an existing iOS iPad app to a Windows 8 app.

¹ The Human Touch: Building Ultrabook™ Applications in a Post-PC Age
From iPad to Win8: The Basics

To understand the design decisions made in this case study to migrate an iPad app to a Windows Store app, it is important to present some of the distinguishing characteristics of iPad and Windows Store apps. The descriptions below are not exhaustive, but give some background for the case study. It is important to highlight other important factors developers need to understand to successfully migrate an app from iOS to Windows 8, such as what developer tools are available, what are the app framework differences, what is the app life cycle for each platform, and which programming languages can be used? The references listed below provide a good starting point in those areas.

The focus of this paper is to discuss both ecosystems from the user experience perspective.

Porting iOS* Apps to Windows* 8 – Overview
Porting App Life Cycle, Settings and Search Features from iOS* to Windows* 8
Enabling Touch in Windows* 8 Style UI Apps with C#
Windows* Style App Development: Using Window Runtime from C#
Resources for iOS Developers (Windows Store apps)
Learn to Build Windows Store apps
Introduction to Creating Windows Store apps using XAML – BUILD Video
Windows Store app Samples
Windows Store and Desktop Code Samples
Ecosystems

Figure 1 shows an iOS (iPad) and a Windows 8 (Ultrabook, tablet, desktop) device side by side. At a glance, the images immediately highlight a few differences that play an important role when designing user experiences for Windows 8: diverse hardware devices (Windows 8 can be installed on tablets, desktops, laptops, Ultrabooks), multiple screen size/resolution combinations, and live tiles (Windows 8 tiles can show dynamic content to attract user’s attention). There are more of course, some of which will be presented further in this paper, however, these examples are easy to gather from the illustrations.
Windows 8 App Structure

Every Windows 8 app is characterized by having a defined set of features and capabilities optimized for touch. Leverage them to create an app that lives consistently inside the Windows 8 ecosystem. Below is a list of the key components of a Windows Store app.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiles</td>
<td>App tiles are the way for users to launch your app. They appear on the Start screen. Tiles can be static or live.</td>
</tr>
<tr>
<td>The App Page</td>
<td>The app page holds all of your UI content and controls. There can be as many app pages as needed. There can also be different views of the same content.</td>
</tr>
<tr>
<td>View States</td>
<td>Windows 8 allows users to work with two apps simultaneously. Users can &quot;snap&quot; their current app so that they can use another app at the same time, or vice versa. Any Windows 8 app can be snapped, so it is important to account for it while designing the experience.</td>
</tr>
<tr>
<td>App Bars</td>
<td>The app bar is the primary command surface for your app after the app page. It is used for navigation elements (typically at the top), commands, and tools (typically at the bottom). App bars are hidden by default. If users need them, they can swipe from the top or bottom edge of the screen to have them appear.</td>
</tr>
<tr>
<td>Charms</td>
<td>Charms are a set of buttons available in every app: search, share, start, devices, and settings. Leverage these charms to perform searches, share content, and define settings for your app.</td>
</tr>
<tr>
<td>Context menus</td>
<td>Use context (pop) menus to show actions that people can perform on text or UI elements in an app. You can use up to five commands on each content menu, like cut, copy, or open with. This limit keeps the context menu uncluttered, easy-to-read, and directly relevant to the text or object that the commands act on.</td>
</tr>
<tr>
<td>Message Dialogs</td>
<td>Message dialogs are dialogs that require explicit user interaction. Use them only when you intend to explicitly ask for user response before continuing to use the app.</td>
</tr>
<tr>
<td>Flyouts</td>
<td>Flyouts are temporary, dismissible dialogs related to what people are currently doing in your app. They are different from message dialogs. Unlike a dialog, flyouts should be used only in response to a user tap or click. Flyouts will be dismissed when the user taps outside them.</td>
</tr>
<tr>
<td>Toasts</td>
<td>When your app is running in the background, it can send notifications (e.g., a new email arrived) to the user via toast notifications. Toasts are good for updating people with information they may want to know right then, but it’s ok if they miss it.</td>
</tr>
</tbody>
</table>
UI Controls

The iOS framework provides a standard set of UI elements, which are the common ground for building iOS apps (see Figure 2). There are also many third-party controls built by other companies to broaden the scope of visual possibilities (see cocoa controls, scoop.it). The iOS Human Interface Guidelines offer detailed information on the actual behavior and usage tips for the standard iOS controls. Similarly, UX Guidelines for Windows Store Apps offer a complete description of the various standard controls available for building Windows Store apps. It is recommended to use the standard controls as much as possible since users are familiar with their appearance and behavior. In addition, standard controls receive automatic updates, thus reducing the need for maintenance by the developer.

**Figure 2:** iPad* COMMON Controls (Source: ICS)
Most of the iOS standard controls have a direct equivalent in Windows 8 (see Figure 3).

<table>
<thead>
<tr>
<th>iOS* Control</th>
<th>Windows* 8 Store Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounded Rectangle Button</td>
<td>Button, HyperlinkButton</td>
</tr>
<tr>
<td>ImageView</td>
<td>Image</td>
</tr>
<tr>
<td>ListView</td>
<td>ListView</td>
</tr>
<tr>
<td>MediaController</td>
<td>MediaElement, SemanticZoom, PasswordBox</td>
</tr>
<tr>
<td>Label</td>
<td>TextBlock (No edit function, with simple text)</td>
</tr>
<tr>
<td></td>
<td>RichTextField (No edit function, with simple text)</td>
</tr>
<tr>
<td>TextField</td>
<td>TextBox (with edit function)</td>
</tr>
<tr>
<td>ProgressView</td>
<td>ProgressBar</td>
</tr>
<tr>
<td>Segmented Control</td>
<td>RadioButton</td>
</tr>
<tr>
<td>Slider</td>
<td>Slider</td>
</tr>
<tr>
<td>WebView</td>
<td>WebView</td>
</tr>
<tr>
<td>Date &amp; Time Picker</td>
<td>Timepicker (WinJS only), Datepicker (WinJS control)</td>
</tr>
<tr>
<td>Picker</td>
<td></td>
</tr>
<tr>
<td>Page Flip</td>
<td>FlipView</td>
</tr>
<tr>
<td>Switch</td>
<td>ToggleSwitch</td>
</tr>
</tbody>
</table>

**Figure 3:** iOS* and Windows* 8 standard controls (Source: ICS)
Multitasking

According to the iOS Human Interface Guidelines, iPad users interact with only one app at a time: “Only one app is visible in the foreground at a time. When people switch from one app to another, the previous app transitions to the background and its user interface goes away.” In Windows 8, however, users can interact with two apps at the same time by means of the Snap view. Figure 4 illustrates this unique behavior. The image shows the Weather and Bing Sports applications side by side. The Weather app is in a Snap view while the Bing Sports is in a Fill view. It is very important to account for Snap views when designing Windows Store apps since users can snap apps at any time. See Guidelines for snapped and fill views (Windows Store apps) to dig deeper in this subject.

**Figure 4:** Two apps running simultaneously on Windows® 8 (Snap and Fill views)
Gestures

In iOS, apps respond only to Gestures, not Clicks. According to the iOS Human Interface Guidelines, “with direct manipulation, users are more engaged with the task.” Although there is truth in that statement for applications optimized for touch, Windows 8 apps face an additional challenge: Since Windows 8 can be installed in a diverse set of devices, some of which might not support touch experiences (e.g., a desktop computer, traditional laptops), apps need to also be accessible via keyboard/mouse combos as well.

The New keyboard shortcuts for Windows 8 and Windows RT article lists all the keyboard shortcuts reserved for various tasks, particularly to complement the touch gestures on various use cases (e.g., desktops or laptops without touch input).
Content vs. Chrome

In iOS, “Controls should look tappable. iOS controls, such as buttons, pickers, and sliders, have contours and gradients that invite touch.” Windows 8 on the other hand, advocates for a different approach in which contours, gradients, additional decorations around interactive elements are considered “chrome” and should be avoided as much as possible. Figure 6 shows part of the Music app. Note the absence of gradients, shadows, or any additional visual effects around the interactive elements of the app. Buttons are flat yet effective in portraying the proper message to the user as to what their functionality is. This approach to design in which the content is the center of the experience and a user interface that looks clean, uncluttered, simple and without too much chrome is consistent with the Signal-to-noise ratio and Horror Vacui principles of design. See Universal Principles of Design for detailed descriptions of these principles.

**Figure 6:** Windows*8 Music app (Source: ICS)
App Navigation, Commands, and Actions

iOS apps provide four different types of bars for various purposes: status bar, navigation bar, tab bar, and toolbar. iOS provides the navigation bar for drilling down through hierarchical content and the tab bar for displaying different groups of content or functionality. A tab bar appears at the bottom edge of the screen. It should be accessible from every location within the app. Figure 7 shows an example of an iOS app illustrating these various controls.

**FIGURE 7**: *iOS* Application Bars (Source: ICS)
Similarly, Windows 8 provides bars for two main goals: commanding and navigation. Application bars are placed at the top and bottom sections of the screen. Contrary to iOS apps, however, Windows Store apps generally have their bars hidden by default. When people swipe from the top or bottom edge of the screen, app bars automatically appear. There are no equivalents to the iOS Status and Navigation bars in Windows 8. Figure 8 shows the location of the Application and Navigation bars in Windows 8.

Figure 8 also shows another important difference between the use of the bars in iOS and Windows 8. On iPads, toolbars appear at the top edge of the screen. In Windows 8 however, actions or commands that apply to the current context should be placed in the Application bar, which is located at the bottom edge. Similarly, in iOS, Tab bars allow people to switch between different subtasks, views, or modes. In Windows 8, these actions are commonly assigned to the Navigation bar, which is located at the top edge. See UX Guidelines for Windows Store Apps for more detailed guidelines on the use of application bars in your app.
App Settings

In iOS, apps have two ways for presenting preferences to users: display preferences inside the app or use the Settings app via a Settings bundle (which is the recommended approach). However, using the Settings bundle means users must switch from their current app to get into Settings to access app preferences.

Windows 8 provides a unified way to set app preferences via the Charms bar (see Figures 10 and 11). This bar is available universally inside the system. Users can invoke it at any time by doing a quick right-to-left flick gesture at the right edge of the screen or by using the “Windows Key+C” keyboard shortcut (if a keyboard is available). See the Activating Windows 8 Contracts in Your App article for more details.

**Figure 8:** Windows®8 Charms Bar

**Figure 9:** Accessing the Xbox® Music app settings via the Charms bar.
Searching Content

iOS provides a Search Bar control that is generally embedded in the toolbar to allow users to perform searches within the app. In some circumstances, however, search icons can also be placed at the bottom Tab bar to enter a “search mode.” For context sensitive searches, search bars are typically located above the list controls displaying the indexed data. Figure 12 illustrates the use of local vs. global searches in iOS.

Similar to Settings, the recommended approach to conduct searches inside a Windows Store app is to use the Search charm, available in the Charms bar (see Figures 10 and 11). By leveraging the charms bar, there is no need to use valuable screen real estate to place search controls. Search results can be customized to display any level of granularity desired or required by the app. Figure 13 illustrates the use of the Search contract to find weather information for the city of Boston. See Activating Windows 8 Contracts in Your App for more details.
Figure 11: Windows® 8 use of the search contract inside the Weather app
Sharing Content

iOS apps allow users to share content via something called the Activity View Controller. In essence, this controller holds all the services/options a user can perform at any point within the current content. The services can be system provided—such as Copy, Twitter, and Print or customized by the app creator. According to the iOS Human Interface Guidelines, a common way to use an Activity View Controller is to allow users to post content they’ve selected to a social media account. Figure 14 shows an example. For the purposes of sharing content, the Activity View Controller works in conjunction with a Share button (generally placed on the toolbar), which reveals the contents of an Activity View Controller.

![Figure 14: iOS Activity View Controller and Share Button (Source: ICS)](image)

Similar to Settings and Search, the recommended approach to share content inside a Windows Store app is to do it via the Share charm. Figure 15 illustrates the use of the Share contract to share the current content with social media accounts.
Figure 13: Windows 8 use of the Share contract inside the Travel app
Content Layout

A very common approach used in iOS iPad app design is to use Split and Table views to lay out the content users will be interacting with. Certainly, many apps do not follow this design pattern, but Split and Table views come standard in iOS and are commonly used. Figure 16 shows an example of this “Master-Detail” view on an iPad. In landscape, both Master and Detail are visible. In portrait, however, the Master is shown in a transient pop-over.

The Porting Advanced User Interfaces from iOS* To Windows 8* article provides a good example that shows how an iOS app that uses a Split and Table view can be moved to Windows 8.
Case Study: A Work Order Management App

Background

Blue Dot is a global enterprise mobile software product and services company that delivers Mobile Field Service solutions to a range of Field, Fleet, Facility, and Factory-based organizations within asset-intensive industries. One of their flagship products is Cyan Work*, which is described as a “comprehensive mobile application used to extend a wide array of work and asset management tasks from leading ERP / EAM systems.” The solution aims to reduce labor by eliminating the need for printing and distributing paper-based work and service orders. It also replaces manual key entry of paper-based work orders into any backend ERP / EAM system, with automatic wireless synchronization. Figure 17 shows the core features of the Cyan Work app.

<table>
<thead>
<tr>
<th>Inbox</th>
<th>Sort, filter, track work by department, due and overdue date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Details</td>
<td>Work order header details, user defined and custom fields</td>
</tr>
<tr>
<td>Activity Details</td>
<td>Multiple activities, task instruction checklist, comments</td>
</tr>
<tr>
<td>Equipment Details</td>
<td>User defined &amp; custom fields, meters, work order history</td>
</tr>
<tr>
<td>Documents</td>
<td>Upload &amp; download multiple file types</td>
</tr>
<tr>
<td>Quick Complete</td>
<td>Rapidly capture comments, closing codes, downtime on one screen</td>
</tr>
<tr>
<td>Parts</td>
<td>Search, issue &amp; return online and offline</td>
</tr>
<tr>
<td>Create Ad Hoc Work Orders</td>
<td>Leverage templates to rapidly create work order in the field</td>
</tr>
<tr>
<td>Quick Labor Booking</td>
<td>Book regular time, overtime, multiple entries, corrections</td>
</tr>
</tbody>
</table>

**FIGURE 15:** Home Screen for Cyan Work* app on iPad*
Layout and App Navigation

The iOS app uses a toolbar located at the top to place some of the common and global commands. Some of these commands can naturally go inside the application bar in Windows 8, which is hidden by default, allowing users to focus on the content and let the options appear only when they are really needed. Figure 18 illustrates this transition.

Figure 16: Transition to leverage the Windows* 8 app bar
Another example in which the Application bar can transition nicely is illustrated in Figure 19. The Create Work order Dialog in iOS has the “Submit” and “Cancel” buttons located at the top of the dialog. The Windows 8 dialog has the buttons placed inside the Application bar. Alternatively, a modal dialog could be used (see Figure 21 page 21 for an example).

**Figure 17:** Transition to create a work order
The Windows 8 experience is meant to be visually focused and direct. Instead of having to drill down yet another level to get to the relevant information associated with the task, why not have it readily accessible from the beginning? Figure 20 illustrates an example. The iOS app requires the user to drill down one more level to select a Status Code. The Windows 8 version honors the “visually focused and direct” statement by allowing users to see the various choices for the Status Code without the need for additional taps.

**Figure 18:** *Work Order Transition to Windows® 8 Leveraging the Horizontal Layout*
Figure 21 shows how the Labor, Documents, and Parts sections can help the user focus without having to drill down deeper into the UI. At a glance, users can see not only the total of hours worked, but they can also see a preview of the most recent time entries submitted (same for Documents and Parts). The iOS design requires the user to get into each section to see the details.

**FIGURE 19: USING WINDOWS® 8 MODAL DIALOGS TO TRANSITION DATA ENTRY**
Figure 22 shows another instance in which the navigation can be flattened a bit to have a more direct experience. The iOS design requires the user to drill down one more level in order to select an activity ID and Labor Type. The Windows 8 design flattens these two entries so there is no need for an additional level of navigation. The iOS app also follows a form-based approach for date and time entries (user enters date/time values via the virtual keyboard). The Windows 8 design favors the use of tiles for the time entries. Once the user taps on a tile, a pop up with the list of possible values allowed for that tile will be displayed. Again, the user interacts with the content directly.
View States to Support Multitasking

As highlighted before, in Windows 8, users can work with two apps simultaneously by means of the Snap and Fill views. Ideally, all screens of the app should have a corresponding Snap view but this is not always the case. There are instances in which certain functionality can only be useful in full view. Figure 23 shows the Windows 8 version of the Cyan Work app (home section) in the snapped view State.

![Cyan Work Snap View](image)

**Figure 23**: *Cyan Work* Snap view
Searching

The iOS version of Cyan Work uses a Search bar located at the top to let users search for work orders (see Figure 24). The Windows 8 version leverages the Search contract, which results in a cleaner, simpler canvas.

**Figure 22: Searching work orders**
Settings

Logging out of the application is generally an infrequent task. There is no need to waste valuable screen real estate on such operations whenever possible. The iOS version of Cyan Work has a Logout button located at the top app bar (see Figure 25). For the Windows 8 app it is recommended to place logout type operations inside the Settings contract. This results again in a cleaner, simpler canvas, which focuses on the content.

**Figure 23: Leveraging the Settings contract to place logout and app options**
References

Activating Windows 8 Contracts in Your App

Blue Dot Solutions, Cyan Work
http://www.bluedotsolutions.com/solutions/cyan-work/

Designing UX for Apps
http://msdn.microsoft.com/library/windows/apps/hh779072/

Cocoa Controls
https://www.cocoacontrols.com/platforms/ios/controls

Enabling Touch in Windows 8* Style UI Apps with C#

Free Windows 8 Wireframe Templates for PowerPoint
http://designmodo.com/windows-8-wireframe/

Horror Vacui

iOS Third Party Developments
http://www.scoop.it/t/ios-third-party-developments

Guidelines for Snapped and Fill Views (Windows Store Apps)

Introduction to Building Windows 8 Apps using XAML – BUILD Video
http://channel9.msdn.com/Events/Build/2012/3-116

iOS Human Interface Guidelines

Learn to Build Windows Store Apps
http://msdn.microsoft.com/library/windows/apps/
New Keyboard Shortcuts for Windows 8 and Windows RT
http://windows.microsoft.com/en-us/windows-8/new-keyboard-shortcuts#1TC=t1

Porting Advanced User Interfaces from iOS* To Windows 8*

Porting App Life Cycle, Settings, and Search Features from iOS* to Windows* 8

Porting iOS* Apps to Windows* 8 – Overview
http://software.intel.com/en-us/articles/porting-ios-apps-to-windows-8-overview

Porting iOS Apps to Windows 8 (2): User Interface
https://mail.google.com/mail/u/0/?ui=2&shva=1#inbox/13dfb55c94b3dfe2

Porting iOS* Custom Charts and Graphs to Windows* 8 UI

Resources for iOS Developers (Windows Store Apps)
http://msdn.microsoft.com/library/windows/apps/jj945493

Signal to Noise Ratio
http://www.vanseodesign.com/web-design/signal-to-noise-ratio/

The Human Touch: Building Ultrabook™ Applications in a Post-PC Age

Universal Principles of Design, Revised and Updated: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions and Teach through Design
http://www.amazon.com/Universal-Principles-Design-Revised-Updated/dp/1592535879

Windows Store App Samples
http://code.msdn.microsoft.com/windowsapps/site/search

Windows* Style App Development: Using Window Runtime from C#
About the Author

Jorge Toro is a User Experience Engineer at Integrated Computer Solutions (ICS). He is currently a team leader within the Microsoft Design Assist Program at ICS, designing experiences for Windows* 8 and Windows Phone 8. He is also actively involved in other consulting projects ranging from software development to user experience design. His professional interests are in the areas of User Interfaces, User Experience, Information Architecture, Human Computer Interaction and all other fields associated with the creation of usable, efficient and effective user interfaces for people. Before joining the team at ICS, Jorge worked as a Software Engineer at VT MAK in Cambridge, MA, designing and implementing simulation software for urban environments. Jorge has also lectured undergraduate and graduate level courses in Human Computer Interaction, Computer Science and Computer Graphics. He holds a BS and MS in Computer Science, a MS in Human Computer Interaction and a Ph.D. in Computer Science with concentrations in Software Engineering, Human Computer Interaction and Computer Graphics... Jorge is a sworn enemy of fruit flies!
appearing herein remains the property of its respective owners, who reserve all rights to such material. No use of any trademarked or copyrighted material shall imply endorsement by the owner of such material.