Porting Windows 7* Applications to Windows 8* Store Apps

Overview

This article will guide developers in the process of porting a Windows 7 application to a Windows 8 Store app. We will discuss the design considerations, language options, and more.

Contents

Overview .................................................................................................................................................. 1
Contents .................................................................................................................................................. 1
Why Convert? ....................................................................................................................................... 1
  Windows Store ................................................................................................................................... 1
  Windows 8 UI Experience .................................................................................................................. 2
  Windows 8 Runtime APIs .................................................................................................................. 3
Porting .................................................................................................................................................. 3
Touch .................................................................................................................................................... 3
Development ........................................................................................................................................ 5
  MFC /Win32* C++ to Windows Runtime C++ .................................................................................. 6
  HTML to Windows Runtime HTML/JavaScript App ...................................................................... 7
Conclusion ........................................................................................................................................... 7
Acknowledgements .............................................................................................................................. 7
About the Author .................................................................................................................................. 7

Why Convert?
One compelling reason to create and port applications to Windows 8 is that users will like them more when designed with a touch-first experience in mind. Touch is arguably more natural for certain types of applications, and these applications make good candidates for being ported to Windows Store apps. Some of the types of applications that are well-suited for conversion include:

- Casual painting and drawing applications
- Games – casual games and hardcore games can both benefit from touch
- Content viewing applications – touch-based interaction with the content for zooming, scrolling, etc.

Why would you want to convert a Windows 7 desktop application to a Windows Store app? There are many potential reasons for doing this:

- Your application will likely provide a more immersive user experience after enabling it for the Windows 8 touch-based UI
- Windows Store is a new source of monetization for your app
- Windows UI look and feel is clean, intuitive touch-based UI
- Access to Windows 8 Runtime APIs, which are new unified APIs for touch, sensors, etc.
- Support of the latest HTML5/CSS3 features for building HTML/JavaScript*-based store apps

Let’s take a closer look at some of these compelling reasons.

**Windows Store**

The store is integrated into the Windows 8 experience. Launched directly from the Start Menu, a familiar tile-based look and feel for browsing and downloading applications quickly.

Here is some more information about using the Windows Store to sell your apps:


**Windows 8 UI Experience**

Touch first, less chrome, fast, fluid; these are just some of the descriptive words used to describe the Windows 8 look and feel. The interface is optimized for mobile form factors and designed to work with finger, mouse, or keyboard input giving users even more ways to interact with their PCs.

The start screen provides a new feature called Live Tiles that gives developers an interesting opportunity to get their app noticed. Live content can be displayed on the tile icon and keep the user updated as to what is new in your application at a glance.

There are even more exciting Windows 8 UI features that offer developers opportunities to create compelling Windows 8 apps. Refer to these articles for more on Windows 8 UI features:

- [http://blogs.msdn.com/b/b8/archive/2012/05/18/creating-the-windows-8-user-experience.aspx](http://blogs.msdn.com/b/b8/archive/2012/05/18/creating-the-windows-8-user-experience.aspx) - Creating the Windows 8 user experience

**Windows 8 Runtime APIs**

New for Windows 8 is a set of APIs called the Windows Runtime. The Windows Runtime is a unified set of APIs that can be used in .Net languages, C++, and HTML/JavaScript equally. The Windows Runtime is similar to using the .NET APIs you may be familiar with, but they have been implemented with COM in C++ to allow use from languages other than .Net. The major differences from traditional .NET APIs include the removal of many of the redundant, obsolete, and legacy APIs.

**Porting**

Now that you have decided to move your application to Windows 8, what is it going to take? The following sections describe some of the options and challenges in moving to Windows 8.

**Touch**

Adding touch support to your Windows 8 application is an important aspect of the porting process.

Let’s look at an example of what type of UI in your existing app would probably be ready for a touch-based interface and what would need some work. Here is a screen shot of a popular painting program:

![Figure 1 - Main drawing page screen shot using Paint from Microsoft Windows 7.](image)

Figure 1 - Main drawing page screen shot using Paint from Microsoft Windows 7.
Highlighted in red are some of the immediately visible problem areas when thinking about how this application will work with touch.

- The title and status bars at the top and bottom - Screen real-estate is at a premium on smaller form factors, the space taken up by these status bars is being wasted since the primary purpose of this application is to paint on the canvas.
- The controls on the menu bar - These small icon controls will be hard to select if only using touch.
- Color selection - The small icons are a problem along with more screen taken up for colors than anything else besides the canvas. A possible design consideration would be to think about how often a user switches colors, is this something that needs to always be available on the screen?

Let's now take a look at what a similar application looks like using the Windows 8 Store UI. This is a similar painting program in two modes. The first figure shows the main canvas drawing part of the application. The second image shows what you see when the settings menu is brought up.

![Image](image-url)

*Figure 2 - Main canvas screen shot using Microsoft Fresh Paint from Windows 8*

Immediately visible is how the canvas is center stage. The only visible UI besides the canvas are the undo and redo arrows. The settings UI is not immediately visible, but using touch or even a mouse will bring up the menu. A right-click or using your finger to swipe up from the bottom brings up this menu and is shown below:
All the same controls are there, you can change your brush, update the color, etc., but now the interface to do this is quite a bit easier if you are using touch. Comparing just the color selection UI, you can tell the buttons are laid out farther apart making it easier to select with just a fingertip. The indicator of what color is selected is also an interesting detail and calls out the color subtly. The status bars that are persistent in the Windows 7 version are there once the settings menu is brought up so nothing is lost.

Here are a couple of great articles for learning more specifics about what makes a good touch-enabled UI:


**Development**

Now that we have looked at some ideas regarding the UI and touch considerations, we can look at how to create this new great looking application.

One nice feature of developing a Windows Store app is the amount of choices available for creating this new application. Possible choices include:

- XAML and C++, C# or Visual Basic*
- HTML5, CSS, and Javascript
- C++, DirectX*
- DirectX, XAML, C++
- A combination of C#, Javascript, and C++

Ideally, your Windows 7 application will already be written in one of these languages or at minimum one of these languages will be familiar to you.

**MFC /Win32* C++ to Windows Runtime C++**

Some of the available APIs that you might have previously used are gone, one example is GDI. There are two main ways to draw things on the screen. One option is to use DirectX and Direct2D* to do your drawing. The second option is to use XAML, which is new for C++ developers. The third option is to use both DirectX and XAML together; this is really a great option to quickly build things like settings and title menus with XAML but keep the main application drawing components as DirectX. A great example of this model is the Fresh Paint application shown in Figures 2 and 3. The menu for selecting paint brushes and colors is designed with XAML and the main canvas of the application is a Direct2D surface.

In addition to using Windows Runtime APIs from C++, some MFC APIs are still allowed but many APIs are not allowed if the app is to be a certified Windows Store app. Some obvious examples of disallowed APIs include things like message boxes, serial port APIs, DirectShow*, etc.

For more specifics on which Win32 APIs can be used in addition to alternative APIs available to Windows apps, see the following articles:


Here are some other resources to help you get started and learn the ins and outs of Windows Runtime C++ development:


**Desktop .Net to Windows Runtime .Net**

The set of .Net APIs available in Windows Store app development is not the same set available when creating Windows 7 apps. Check out the following Microsoft article [.NET for Windows Store apps overview](http://msdn.microsoft.com/en-us/library/windows/apps/hh700360) for the specific details on what has changed. Obvious changes include using a folder picker class that integrates into the Windows 8 UI instead of the FolderBrowserDialog used for desktop folder browsing. The article [Windows* Style app development: Using Window Runtime from C#](http://msdn.microsoft.com/en-us/library/windows/apps/hh700360) provides some further details on the nuances of using C# from Windows 8 Runtime.
HTML to Windows Runtime HTML/JavaScript App

With the addition of HTML5/JavaScript language being an option as a first-class development environment, the opportunity to move existing HTML-based web sites and apps to Windows Store apps is possible.

Much of your HTML code will be reusable when moving to a Windows Store app, but be aware of these considerations:

- Enabling support for touch and gestures.
- Enabling support for multiple screen sizes, although this problem already exists when designing for different computer monitors. The Windows 8 snap view is another screen size to consider when figuring out how your application will scale down.
- Plugging into the Windows 8 App framework. This includes framework features such as the application lifecycle, contracts, live tiles, sensors, etc.
- Conforming to the Windows 8 UI style. There are libraries available to make this as simple as possible.

Here is some further reading on HTML5 and porting to an HTML Windows Store app:

- [http://blogs.msdn.com/b/ie/archive/2012/05/31/windows-release-preview-the-sixth-ie10-platform-preview.aspx](http://blogs.msdn.com/b/ie/archive/2012/05/31/windows-release-preview-the-sixth-ie10-platform-preview.aspx) – A blog on the HTML5 engine that powers IE10 and Windows 8 Style HTML Apps

**Conclusion**

Microsoft has done a great job leveraging the languages used most for Windows 7 desktop development to make the transition to Windows Store apps as easy as possible. With some thought around how to make use of Touch and Windows 8-specific features, your Windows 7 application will greatly benefit from being ported to a Window Store app.

**Acknowledgements**

Thanks to Ashok Emani for contributions to this article.

**About the Author**

Nathan Totura is an application engineer in Intel’s Software and Services Group. Currently working on the Intel® Atom™ processor-enabling team, he helps connect software developers with Intel technology and resources. Primarily these technologies include tablets and handsets on the Android®, Windows 8, and Tizen® platforms.
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel, the Intel logo, and Atom are trademarks of Intel Corporation in the US and/or other countries.

Copyright © 2012 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.
This sample source code is released under the Intel Sample Source Code License Agreement