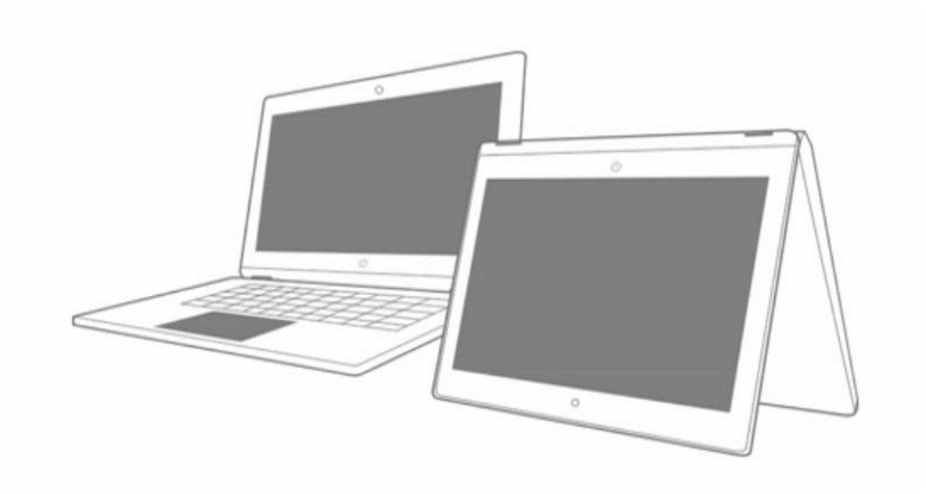
Designing for Ultrabook™ Devices and Touch-enabled Desktop Applications

With the introduction of Windows 8* and touch-enabled computers, like Intel Ultrabook™ devices, where touch is an additional input method, users have new ways of interacting with applications and software. Historically while touch has been limited to niche devices requiring specialized software, today we're seeing a lot of manufacturers creating devices supporting these new input methods.

The Ultrabook is unique in that it includes not just touch, but also has a keyboard and mouse trackpad in a laptop format for traditional usage. So users have the option of using the keyboard as they have in the past, but can also use touch as a source of input.

The broad adoption of smartphones and tablet computers means designers need to adjust their mindset to be aware of the capabilities and limitations of touch. This document will look at how this impacts the design of user interfaces and provide some guidelines for building great software that will encourage



interaction and a great user experience.

Touch

Touch input is not entirely new. It's actually been around for quite some time (the first touch-based application I worked on was in the early 1990's), but has only recently become affordable to include in mainstream devices. Where it was once rare to build touch-enabled applications, today's smartphones are almost all touch-based, and end users are embracing them. With the release of Windows 8 built from the ground up as a touch first system, we can expect more devices to include it, whether as a tablet, desktop, or laptop configuration.

Users should be able to interact with the Ultrabook's touch screen in a natural way. If there's a button on the screen, they should be able to reach up and tap it and have it behave as if they moved the mouse over it and clicked it. Similarly, gestures like pinch to zoom out or in, and touch and drag to pan the content around the screen should behave in such a way that the action has the expected reaction.

Touch is a great addition to the capabilities of a fully featured machine.

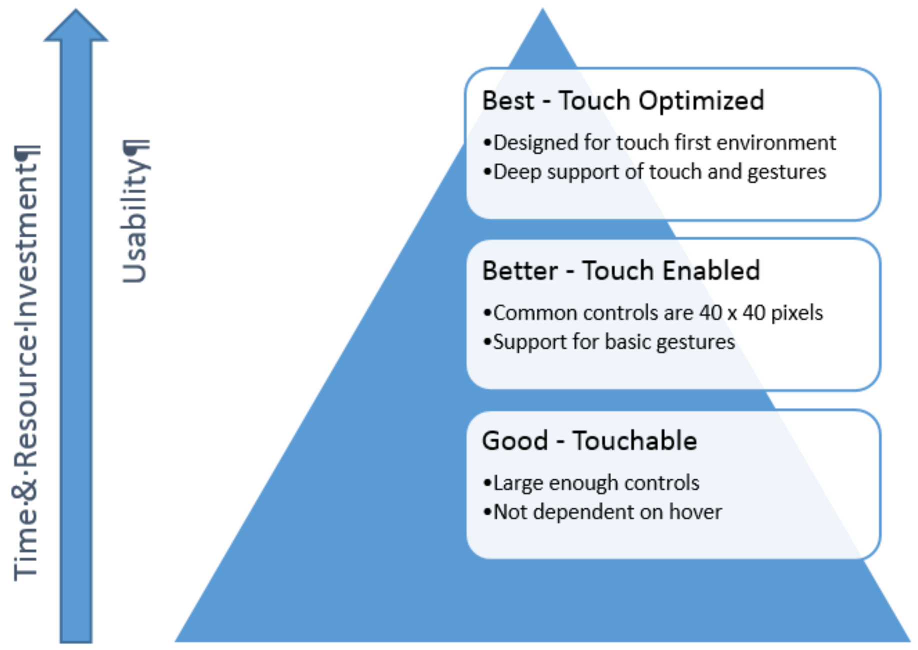
Is it touchable

Users are not only in touch-enabled situations, but in some cases, such as when a device's keyboard is hidden or removed, may be in a touch only environment. If an application depends on mouse behaviors such as hover to reveal commands or actions, users will not be able to use the software. To make it useable, we have to ensure that the controls are large enough to support touch and that the placement makes sense given the expected usage patterns.

Designing for touch includes determining how much investment of time and resources makes sense, balanced against the importance to the business and the return on investment (ROI). For example, investing more in a customer-facing application will likely result in a higher ROI, than investing in an internal application used by just a few users (although not true if it's critical to the business).

Microsoft defines several categories of touch-enabled applications. These include:

- **Touchable**: Controls are sized large enough to function appropriately in a touch only environment, which research has shown to be a minimum of 23 x 23 pixels (6 mm) and have sufficient room between them to prevent accidental selection. The application is aware of and uses system gestures like flick and pan to move scrollable regions and does not require use of hover.
- **Touch-enabled**: Most frequently used controls are at least 40x40 pixels (10 mm), and the standard relevant gestures of touch targets are supported such as panning, pinch and zoom, and rotation. Basic multi-touch manipulations are implemented and behave as expected.
- **Touch-optimized**: The content has been revisited to minimize clutter and maximize user experience. Tasks are performed using controls strategically placed in easy-to-access areas on the screen. Higher level interaction features like inertia and momentum are used to make the


Time·&·Resource·Investment¶
Usability¶
Best - Touch Optimized
•Designed for touch first environment
•Deep support of touch and gestures
Better - Touch Enabled
•Common controls are 40 x 40 pixels
•Support for basic gestures
Good - Touchable
•Large enough controls
•Not dependent on hover

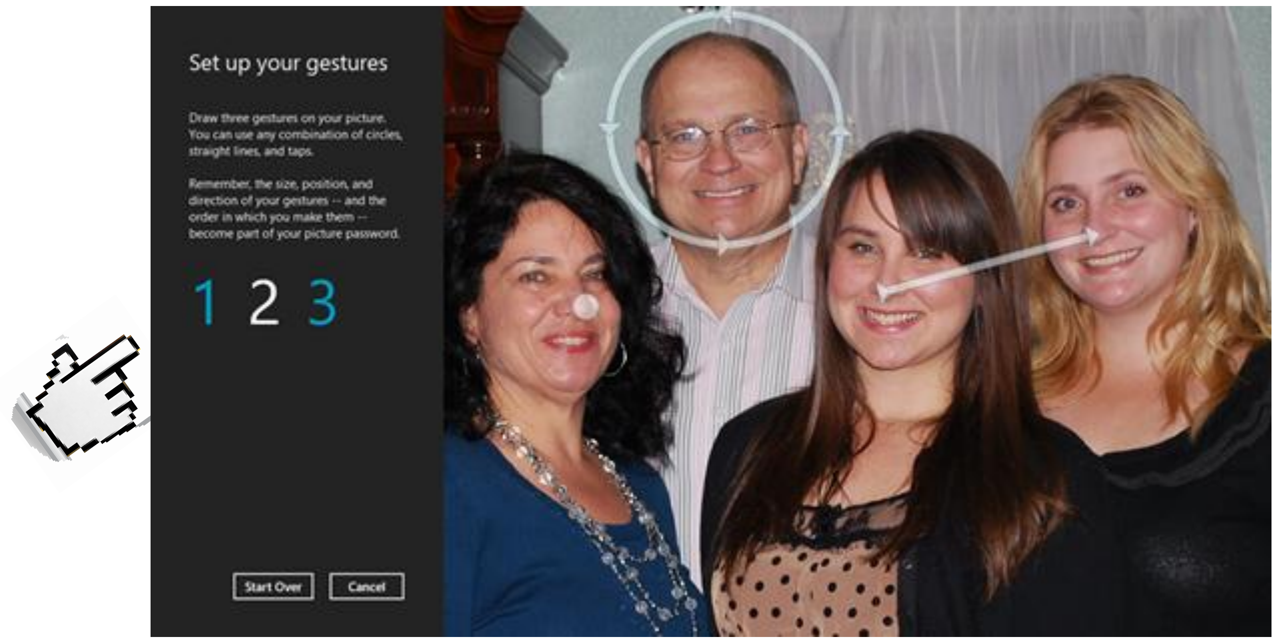
interface come to life.

The challenge of migrating existing applications to this new environment is to make sure they are, at minimum, touchable in the sense that they will function in a touch only environment. To help you make decisions about how much investment is appropriate, it helps to look at some of the design principles used in redesigning the operating system for touch, and see how they apply to the desktop.

Touch Design Principles

The designers of Windows 8, introduced at the Build conference in 2011, discussed the set of principles they used to develop the framework for making touch a first class citizen. These include:

Touch should be natural and intuitive. Touch input is captured by the user touching the screen to select and manipulate objects and controls. This means that the end user does not require special training to interact with the application. For example when using the Ultrabook, one of the great features is you can create and use a touch password to log in. Not only is it more personal, it is faster and more secure than some of the other modes of authentication.


Set up your gestures

Draw three gestures on your picture. You can use any combination of circles, straight lines, and taps.

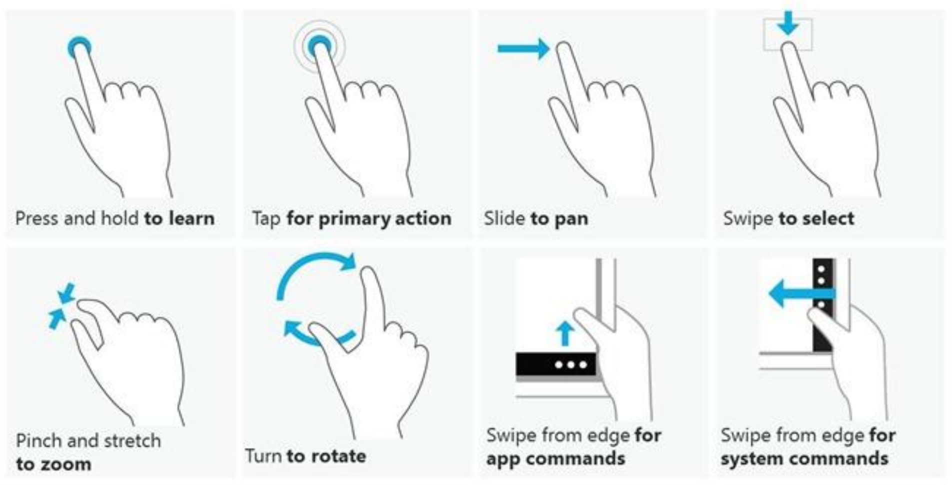
Remember, the size, position, and direction of your gestures -- and the order in which you make them -- become part of your picture password.

Direct and engaging. Touching a control manipulates that control, not something in a different area of the interface, and gives the user feedback that something is happening. This could include sounds or vibration. Including features like inertia and momentum provides a more realistic experience and is more in tune with the real world.

For instance flicking a finger on a control across the screen moves a picture, but it behaves with friction and inertia to come to an eventual rest. If they try to do something that is not supported, like moving a control to a location on the screen that doesn't work, it moves slightly but snaps back into place.

Portable and Consistent. Applications should follow industry standards and implement gestures in a consistent way. Gestures and manipulations mean the same thing across different applications. For example, a Pinch gesture will zoom out or affect the control by resizing it to be smaller while the expand gesture does the opposite. Touch and dragging performs a panning manipulation. You should think twice about creating new custom gestures, especially if one of the system gestures does the same thing.

Not intrusive. The controls that are designed to be touch aware are easy to access and fit in a logical place in the interface design. Manipulation of these objects does not obscure or prevent completion of tasks.

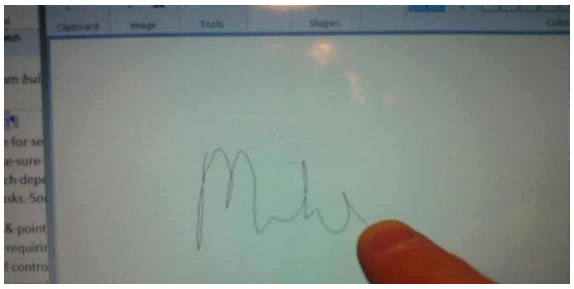

Press and hold to learn
Tap for primary action
Slide to pan
Swipe to select
Pinch and stretch to zoom
Turn to rotate
Swipe from edge for app commands
Swipe from edge for system commands

From building Windows 8 blog - Windows 8 touch language

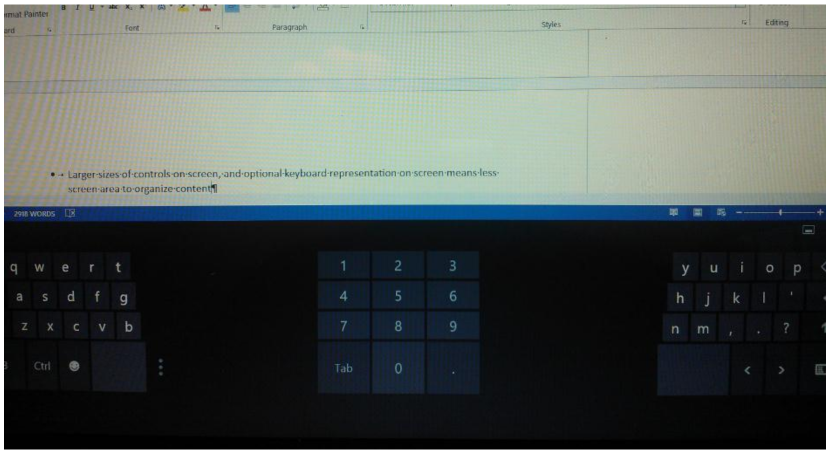
Limitations of Touch

Touch is a less accurate for selection, and events like hover are not available to the application. So the designer needs to make sure controls are large enough, with sufficient margin to enable them to be touched. Controls that depend on hover (like mouse over) to complete actions or tasks should not be incorporated. Some of the limitations to be aware of include:

- While a mouse or pointer is highly accurate allowing the user to select specific pixels, touch is much less so, which requires touch targets to be large enough and separate enough to allow selection. For example when writing on a screen with touch, you can't see where the point of contact is. Other modes for working with textual input may be more effective.



- Larger sizes of controls on screen and optional keyboard representation on screen means less screen area to show content.
- When hover is not available, controls that depend on it will need to be reconsidered.
- Working with text, input and selection are more difficult.
- On screen keyboards leave less space for your content.



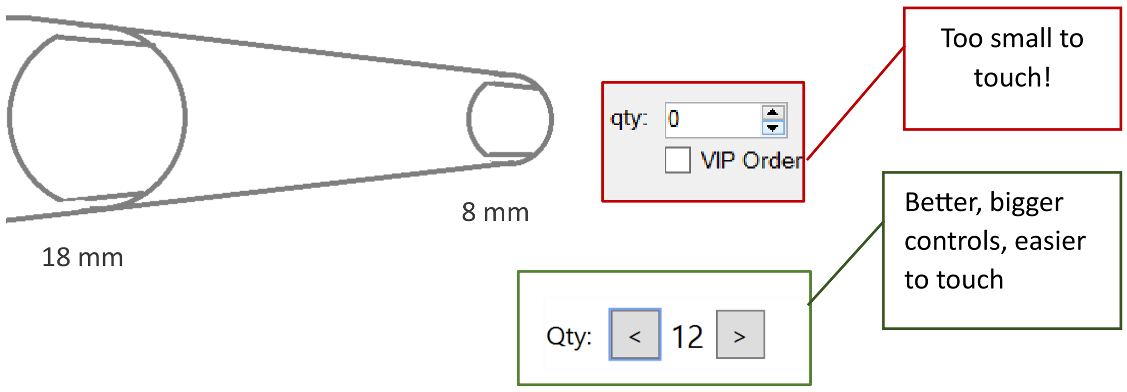
• → Larger·sizes·of·controls·on·screen,·and·optional·keyboard·representation·on·screen·means·less·screen·area·to·organize·content¶

- Touch must be forgiving, in that if the user accidently leans against the screen and touches something, the app shouldn't do unrecoverable destructive behavior.

Touch is intended to be an easy-to-use and natural way to interact with an application, and may be one of many input devices available, or it may be the only input available. This means your application cannot depend on the availability of keyboard and mouse, although and application that has good support for mouse will translate easier into a touch-enabled application because touch and gesture events are translated by the OS into mouse events.

Touch target sizing considerations

One of the first things to recognize about touch as an input is that the selection device—the human finger—is much larger than the pointer on a mouse or pen. Finger sizes range from 8 mm for a child up to 18 mm or larger for a professional athlete. The average size is somewhere between 11 mm and 15



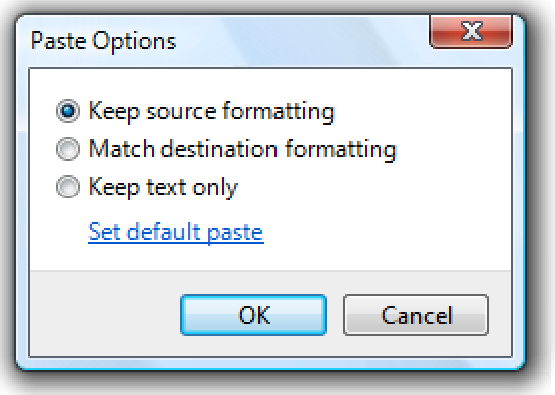
mm.

Guidance

The target sizing for touch needs to be large enough to be touchable, or at least 23 x 23 pixels (about 6 mm). Some controls are too small to be touchable, like a spin control, which is 9 x 15 pixels, so the designer should consider alternative ways to present the user with options.

Spacing between controls is another area where more is better. If links or choices are too close together, the user is more likely to accidently click the wrong option. Therefore, it is a recommended practice to increase the space between controls where space allows.

Acceptable



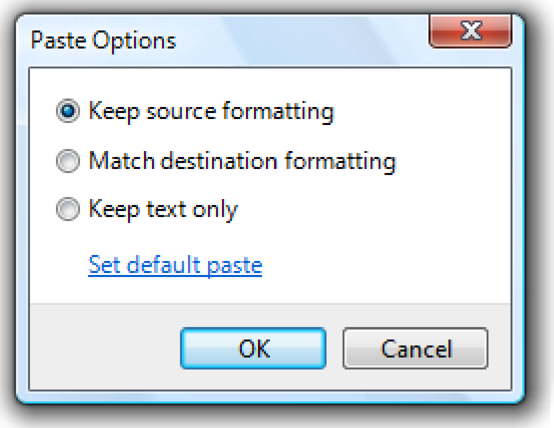
Paste Options

● Keep source formatting
○ Match destination formatting
○ Keep text only

Set default paste

OK Cancel

Better



Paste Options

● Keep source formatting
○ Match destination formatting
○ Keep text only

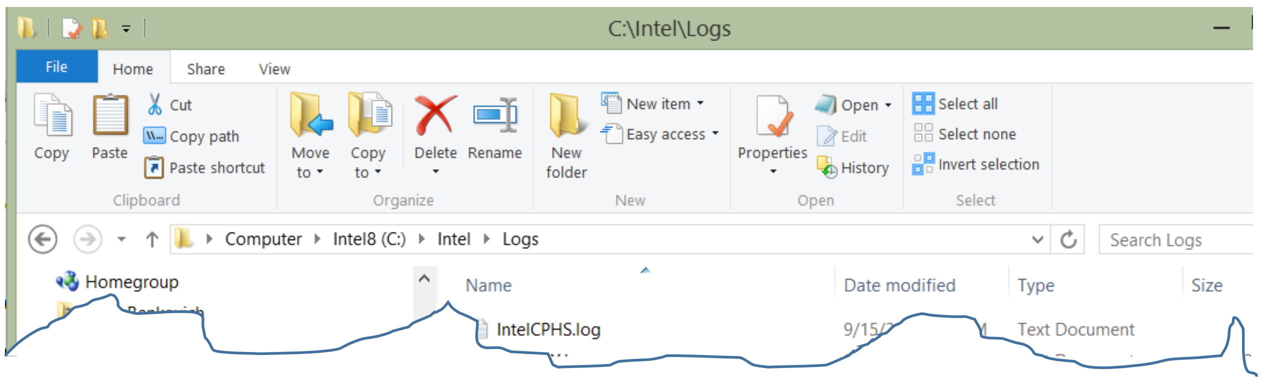
Set default paste

OK Cancel

The default size of toolbar icons (16 x 16 pixels) is hard to touch, so you should look at using larger icons and include enough margin between to make it easier to make the desired selection. A five pixel margin is good, but more is better if your screen layout permits. Additionally consider the types of controls included in the Toolbar, like drop-down lists or text boxes, and make them large enough to function.

Menus are an area where the spacing between commands can be tight for a touch only device. By default there are 7 pixels between commands, but if you add 4 more you will have an easier-to-use menuing system. Fortunately you can leverage the OS native features to help with the consistency of experience and the level of effort required to implement your application.

If these guidelines seem constraining, consider using the ribbon instead. This control, introduced in 2007, is by design touch friendly in that it allows you to choose button sizes and organization to meet very complex scenarios. For example, Windows Explorer uses a standard ribbon to make it easy to find commands for copying or renaming files, as well as provide access to less used commands in an intuitive, easy-to-use way. Notice that the most used commands are represented with larger icons.

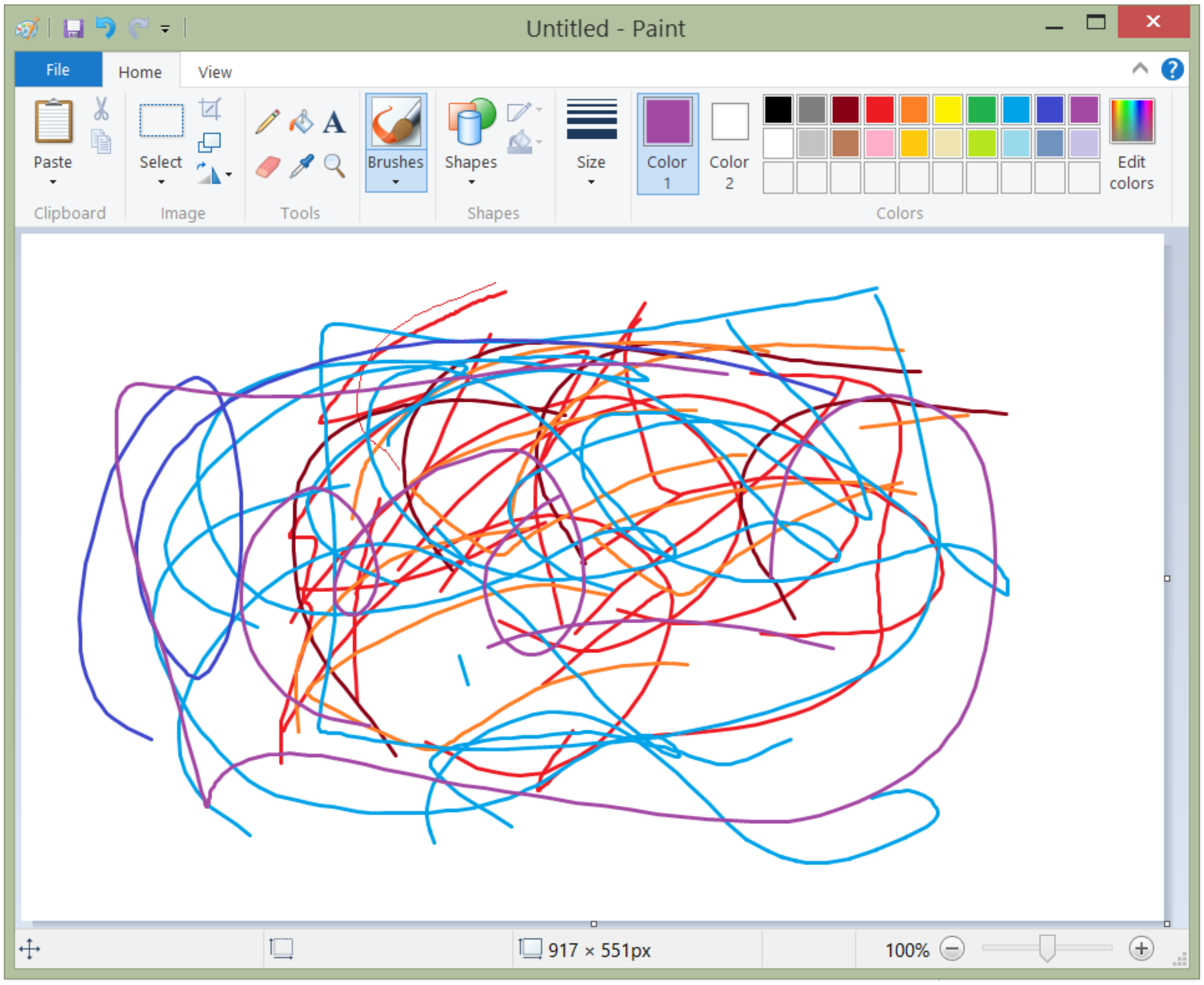


The ribbon is also collapsible. So when it's not needed, the user can hide it from view. Buttons, check boxes, combo boxes, and other commands can be designed into the ribbon making it easier for users to find and complete tasks as needed.

Locality – placement of controls

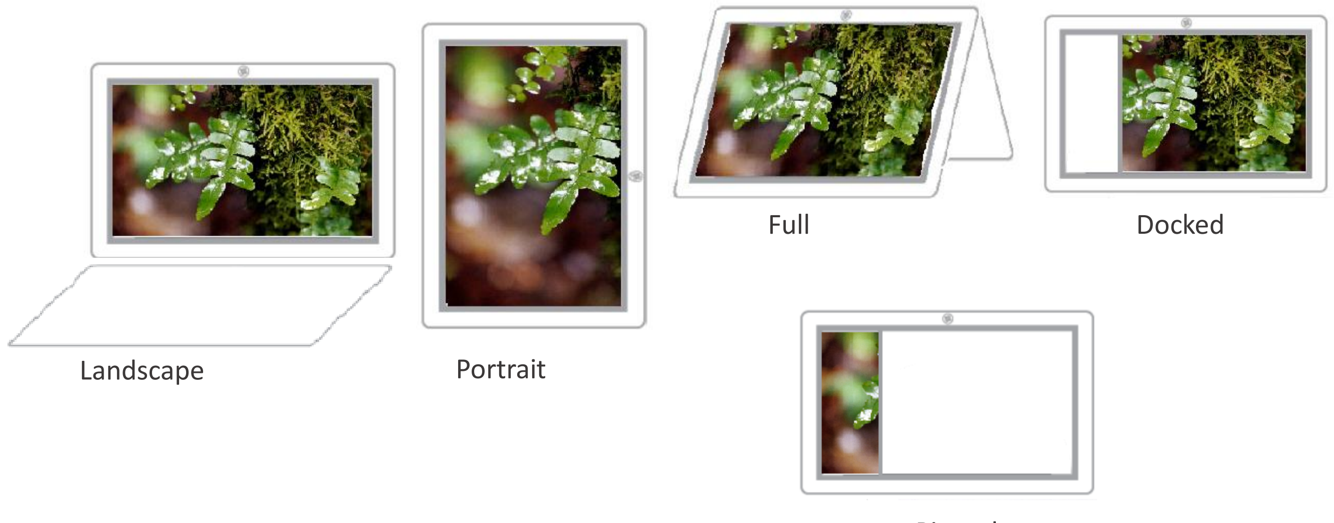
Traditionally developers could depend on hardware limitations to restrict the number of layouts that the application had to be aware of. Standard landscape orientation with a 1024 x 768 screen size meant that an application designer could simply lay out the controls on a canvas, and the app would work for most users. Since tablets and Ultrabooks can be easily rotated, the fixed layout of controls means that certain areas of the screen are going to be easier to touch than others. What works well for placement of controls on a tablet may be the opposite on a laptop.

This is where an adaptive layout makes sense. Applications built with a markup language like XAML can use layout containers to place and keep controls close to the user. Detecting input devices, like the availability of a physical keyboard to determine which layout makes sense, will help your application adapt to the device being used. It's a good idea to make sure that there is a clear differentiation between common controls, both in placement and in the images used. For example you don't want the user to accidentally hit delete or cancel when they meant to hit save.



Designing for multiple layouts

Designers need to ensure that the software will adapt to the environment the user chooses to run it in. This means we need to test a matrix of resolutions to ensure the software behaves well, including testing each resolution against every likely input method.



This increases the number of scenarios to test. But by being aware of the different modes of operation, you will know if your software will be useable for how users will expect to be able to use it.

System Controls and Gestures

One note about Windows 8 is that the margins of the screens are used by the operating system as a common place to access certain features and behaviors. For example, swiping from the right side of the screen will display the Charms. Swiping from the left side allows the user to iterate thru the active application stack. You should avoid placing controls too close to be confused with these system gestures.

- **Charms –** Swiping in from the left edge of screen displays the standard contracts supported by Windows 8 including Search, Share, Start, Devices and Settings
- **Application Stack –** Swiping from right edge of screen switches between active applications. Swipe in and out displays thumbnails of active applications.
- **Application Bars –** Swiping from top or bottom displays an application bar with commands contextually consistent with the active application, although in desktop mode there are no active application bars because the desktop is active.

Be responsive

When implementing a behavior, you should make sure your application responds instantly. Long delays when interacting with an application causes users to wonder if the application understands the gesture or manipulation. Remember that touch should be direct and engaging. So when implementing a behavior, try to offload complex processing to the background where possible.

A recommendation for complex animations is that if they require significant time and/or processing, you could run them from a separate thread. Alternately you could also complete detailed rendering at the end of an animation.

Consider the content

It should be obvious that if we need to have larger control sizes and more margin between controls that we cannot place as much content on a single screen. This is not necessarily a bad thing, in that it forces us to look at what is the core data for the specific point of interaction. Reducing the screen clutter by removing less important details or moving them to other screens and providing a means for smooth navigation between them can improve the usability of an application.

Simplicity is a good thing. The challenge is to identify what is important to keep the focus on the intent of the application. Determining what users need to have on the screen at different points will help prioritize the content. Sometimes context of the currently selected control can help determine what you should show and hide at different times.

Summary

While it is true that most Windows 7 applications will continue to run on Windows 8, it does not mean that the applications will work in the full sense they could. When the application that depends on hover or pixel perfect pointing and selection accuracy are used in a touch only device, the need to look at the design and integration of touch becomes critical.

All applications are different, but you should carefully determine how much investment makes sense. At a minimum, make applications touchable by using controls that are large enough to support a touch only environment. Reduce clutter and use a clean interface that leverages touch principles of size, spacing, and placement of controls.

A final note is that while touch is great, on the Ultrabook it is an additional input to the keyboard and mouse. The maxim "Because you can, doesn't mean you should" holds true in the sense that you should use touch for what it's best for, and the other inputs for their strengths. When designing applications, the form factor and usage patterns should play into how you lay out your content, and there are several videos and articles that highlight studies that have been done (see the references below).

The touch-enabled Ultrabook is a great device that will change how people use laptops. Get started with taking advantage of the new features and explore how to build applications that are touch-enabled.

References

Resources that were used to create this article include the following:

- Article about Daria Loi's study on Touch - http://www.intelfreepress.com/news/do-people-want-touch-on-laptop-screens/
- Steve Chippy Paine's video on Ultrabook Touchscreen Usage Examples - http://www.youtube.com/watch?v=WIm2w2oNdA8&feature=relmfu
- Windows touch guidelines - http://msdn.microsoft.com/en-us/library/windows/desktop/cc872774.aspx
- Design adaptive websites - http://msdn.microsoft.com/library/ie/jj583806.aspx
- Luke Wroblewski's video Re-imagining Apps for Ultrabook - http://software.intel.com/en-us/videos/re-imagining-apps-for-ultrabook-part-1-touch-interfaces
- Learn about Windows Touch - http://msdn.microsoft.com/en-us/library/windows/desktop/touch.aspx
- Build Windows 8 Blog - http://blogs.msdn.com/b/b8/archive/2012/03/28/touch-hardware-and-windows-8.aspx
- Signing in with a picture password - http://blogs.msdn.com/b/b8/archive/2011/12/16/signing-in-with-a-picture-password.aspx