



Microsoft Surface 2.0

Design and Interaction Guide

Principles & Guidelines for Designing and Developing
Surface Applications

July 2011

COPYRIGHT

This document is provided for informational purposes only, and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, financial and other data, and events depicted herein, are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, places, financial or other data, or events is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2011 Microsoft Corporation. All rights reserved.

Microsoft, Microsoft Surface, the Microsoft Surface logo, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

INTRODUCTION

Microsoft Surface enables developers and designers to deliver amazing, social, and highly interactive experiences to their customers. People can use the 360° degree interface from all sides for face-to-face collaboration, cooperation, and building trust. It's intimate enough to facilitate the close back-and-forth of a consultation and makes intuitive sense to people, young and old. Touch. Push. Pull. Turn. Place an object on the screen. Surface creates new opportunities to socially interact with people and content.

Simply put, there is no easier way to bring people together to connect, learn, have fun, and decide. Surface is what computing always promises to be – intuitive, informative, engaging, and helpful – and with it designers and developers create experiences customers will seek out and remember.

Developing compelling Surface experiences requires a different approach to interface design. This document presents design principles and guidelines to address key aspects of application interface design including: interaction, visual, sound, text, and more. Use these principles and practices as a starting point to get the most out of the Surface software and hardware platform's unique capabilities.

Section 2.0:

GETTING STARTED

The following section highlights important things to keep in mind while planning your Surface applications.

Section 2.1:

SURFACE BRAND TENETS

Delightful

Enchanting, surprising, rewarding

Surface rewards people's curiosity in a responsive and captivating way that connects with their senses of touch, sight, and hearing.

Smart

Intuitive, savvy, perceptive

Surface anticipates people's next move. It entices them and reveals key information right when it's needed.

Understated

Elegant, subtle, intriguing

Surface helps people perceive the potential contained within the experience through a thoughtful and tactful approach.

Section 2.2:

KEY CONSIDERATIONS

1. Pixels versus inches

Microsoft Surface hardware exposes the

`Microsoft.Surface.Input.InteractiveSurfaceDevice.PhysicalScaleTransform` property.

This property represents the ratio between 96 dot per inch (DPI) and the logical DPI used by Microsoft Surface units. Using this transform, 96 pixels will result in one real inch on Microsoft Surface displays.

The `LogicalDpiX` and `LogicalDpiY` properties are also exposed. These return the number of pixels that show as one inch on a Surface device.

These properties allow people to write applications where a `UIElement` will have the same physical display size, no matter what type of Surface device is used. These properties help design applications that have content that is properly sized, no matter the screen dimension or resolution. For instance, a keyboard should come up the same size no matter what hardware is used.

Disclaimer: Not all displays expose the real size of a pixel, in that case the default 96 DPI is assumed for that type of hardware.

2. Globalization and localization

Microsoft Surface applications are available in a number of languages and regions around the world. Developers selling their applications to global markets should ensure their applications are world ready. Follow best practices designing the application interface to support varying text string lengths, date formats; and be aware of cultural sensitivities around use of colors, images, and geopolitical issues. MSDN online has a variety of topics dealing with these best practices.

Generally, you should provide at least 40% buffer space for localized strings.

3. Understanding your audience

Prior to Surface application development, it is critical that application developers and designers understand who they are developing for, what setting the application is likely to be used, and the business goals for the application.

Section 3.0:

DESIGN PRINCIPLES

Design is a critical part of the Microsoft Surface experience. Surface applications are based on a special set of principles that helps ensure content is always at the heart of the experience.

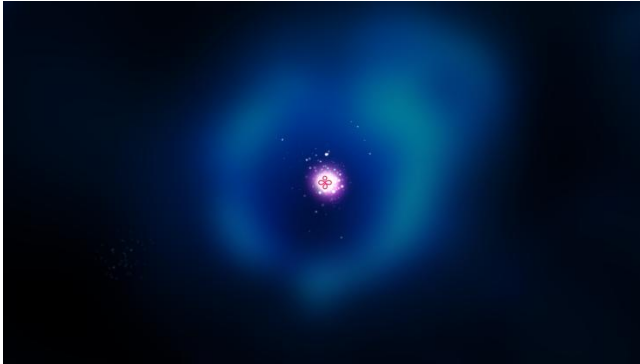
Customer requirements and implementations will vary across applications, so it's strongly recommended that developers use these principles as the foundation for designing and stylizing their applications. Surface design principles should be used as the inspirational

starting point to create consistent and enjoyable experiences that draw people in. Surface application design follows five principles:

01 Simple

Light and open

The Surface experience is never overly decorated or embellished.

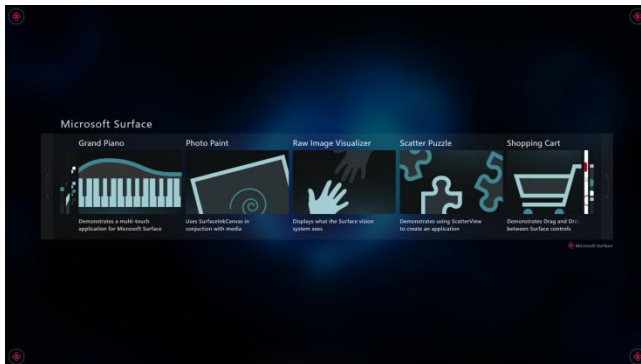


The Begin Experience icon is a simple and straightforward navigation point to begin the Surface experience. In contrast with the playful, organic Attract mode, the Begin icon is graphic and minimal to indicate its function as a navigation point.

02 Organized

Hierarchical and structured

The most important elements have the most visual weight.

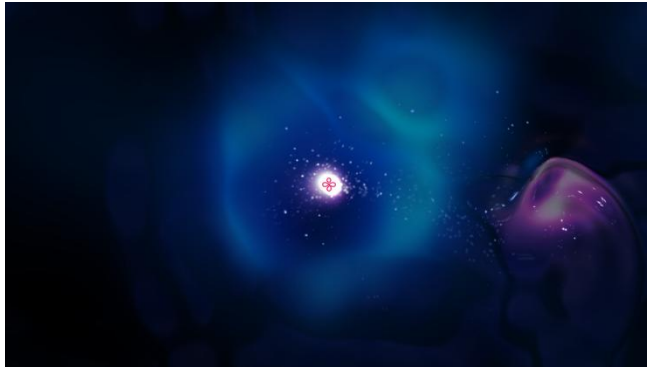


The Surface Launcher is visually organized and structured in a way that places focus on the application icons, titles, and descriptive text. The Launcher container is subtle, and helps maintain a sense of organization when placed over any number of backgrounds.

03 Authentically Digital

True-to-form, honest

Surface uses an honest and direct design style that reflects the digital medium and emulates the physical world only when appropriate.



The Attract state is based on real world physics and water interaction. It adds stylized digital particles and colorful reflections to move beyond what is normally possible in the real world, making the experience true to the digital medium.

04 Content Oriented

Focused and discoverable

Content is always at the core of the experience – controls and chrome are secondary.



The photo content shown here is the most important part of the experience. The Element Menu indicator appears only if a photo is touched. People can then touch the Element Menu icon to reveal a number of interaction options.

05 Lively

Soulful and responsive

Surface is always responsive – animation and motion enhance content and bring it to life onscreen.



Surface transitions bring the experience to life. The application loading transition shown here illustrates the way Surface responds to launching an application; the application icon scales slightly up under the user's finger, the Launcher scales back and dims, and the loading bar appears under the icon to indicate loading progress.

Section 3.1:

INTERACTION DESIGN GUIDELINES

Interaction design defines the interplay of the software experience with people's behaviors, responses, touch, and movement. These interaction guidelines are related to the Surface design principles covered in the previous section, and they move beyond the boundaries of those principles to include an encompassing perspective on creating Surface applications that are intuitive, engaging, and seamless.

There is a great deal of interconnectedness among the principles; but how much emphasis you place on any given guideline depends on the type of application you're creating.

1. Create experiences for several people to use at the same time

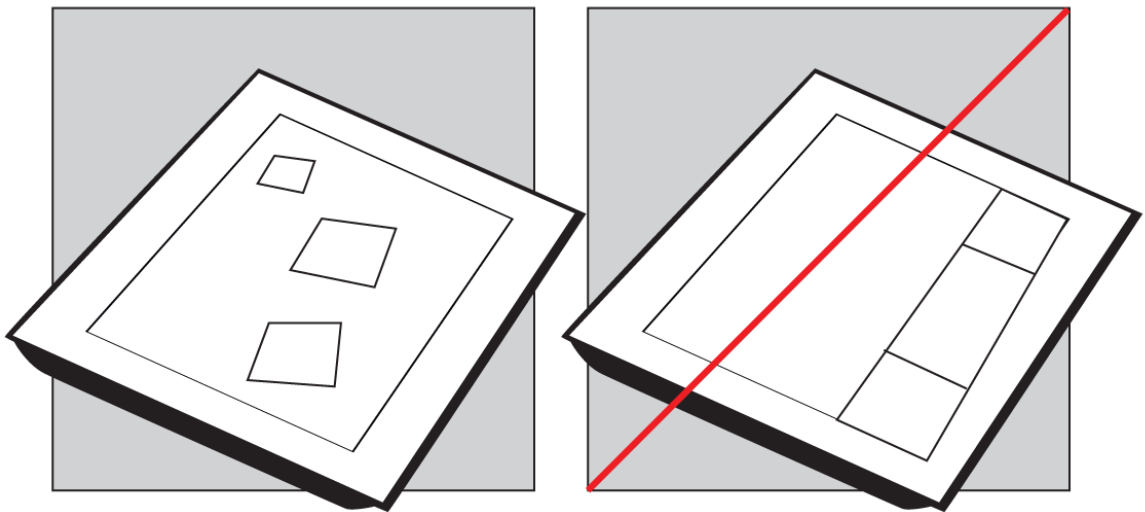
Microsoft Surface recognizes and responds to over 50 different touches at the same time. It sees fingers and objects touching the screen. This enables several people to gather around Surface and share applications, elevating solitary activities to social experiences.



a. Use 360° degree application design for horizontal deployments

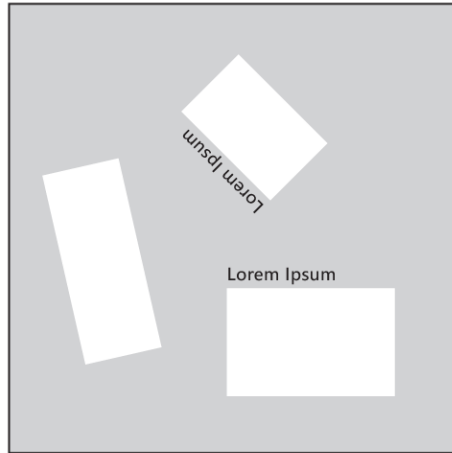
Surface recognizes touch orientation – it sees which direction fingers and special objects are pointed as they contact the screen. This enables developers to generally determine which side of the screen a particular person is on. People will use Surface from all sides so it's important that horizontal Surface deployments be designed for 360° usage.

- Make sure everyone can use content, avoid facing it toward only one side of the display.

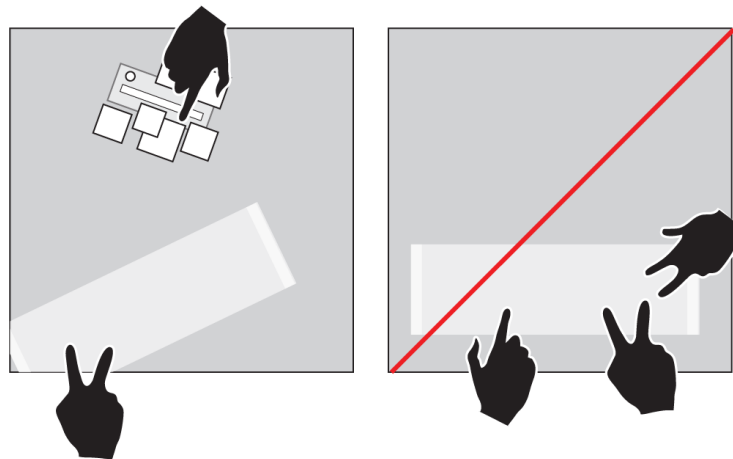


- Enable people to change the direction content faces. Don't lock content to a particular screen direction.

- Use the ScatterView control provided in the Microsoft Surface 2.0 SDK to quickly enable 360° degree experiences.



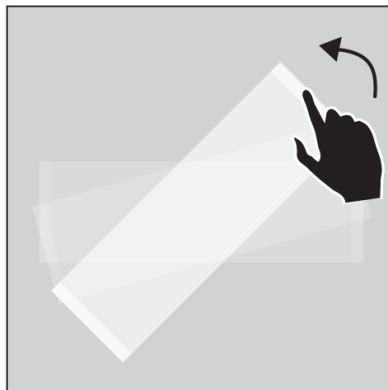
- Make sure everyone can reach and use key application elements from all sides of the unit.



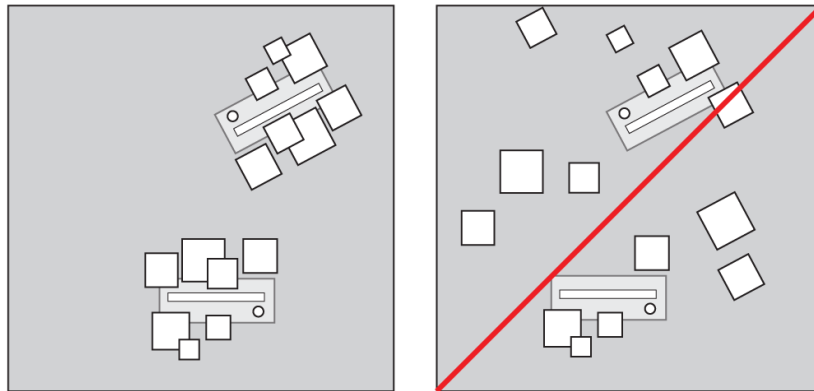
- Some customer application proposals may require people to be on specific sides of the screen. This generally limits application usability, so consider it carefully.



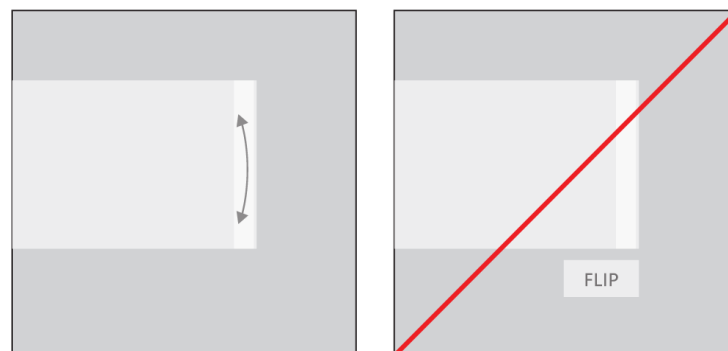
- For example, a collaborative sales application may require a salesperson to always be on one side of the unit and the customers to always be on the opposite side. While this may seem to meet the initial customer need, it will diminish the application as it limits its social appeal.
- Ensure the interface is truly accessible from all sides of the Surface unit. Start using and testing the application in various orientations from the start of application development. Make sure many people share the application at the same time throughout the testing process.
- If the application is freely oriented (has no top or bottom), enable people to change content orientation within the application.



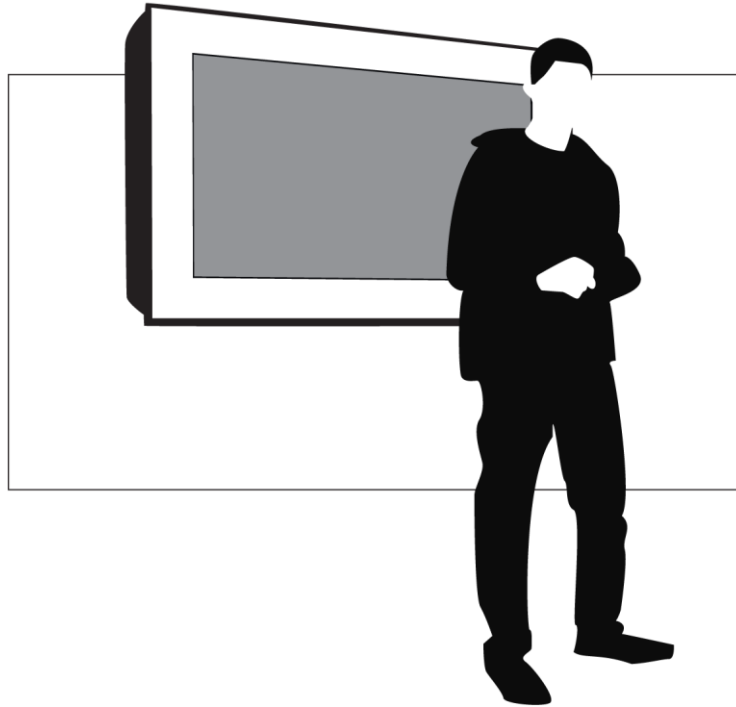
- Surface applications must face new content and interface elements to the people using it. Developers can use finger orientation and the direction of activated content and controls to make reasonable assumptions about which way to point new content and elements.
- If the application still has a distinct top and bottom, give people a way within the application to change the orientation of the entire application.



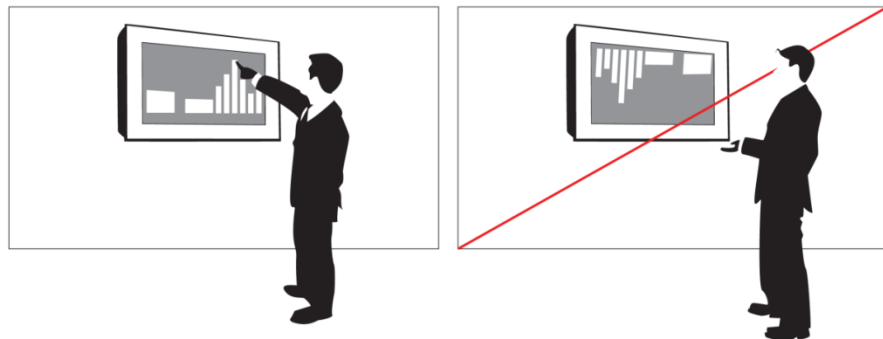
- Simple “flip” buttons are prone to people accidentally touching them, ruining the experience. The Surface Launcher uses a side handle which requiring people to perform a simple, deliberate, action before the grouped content is rotated.



- b. Use 180° degree application design for vertical deployments



- Orient new content and interface elements toward the bottom of the screen.
- Make sure new content appears right-side up.

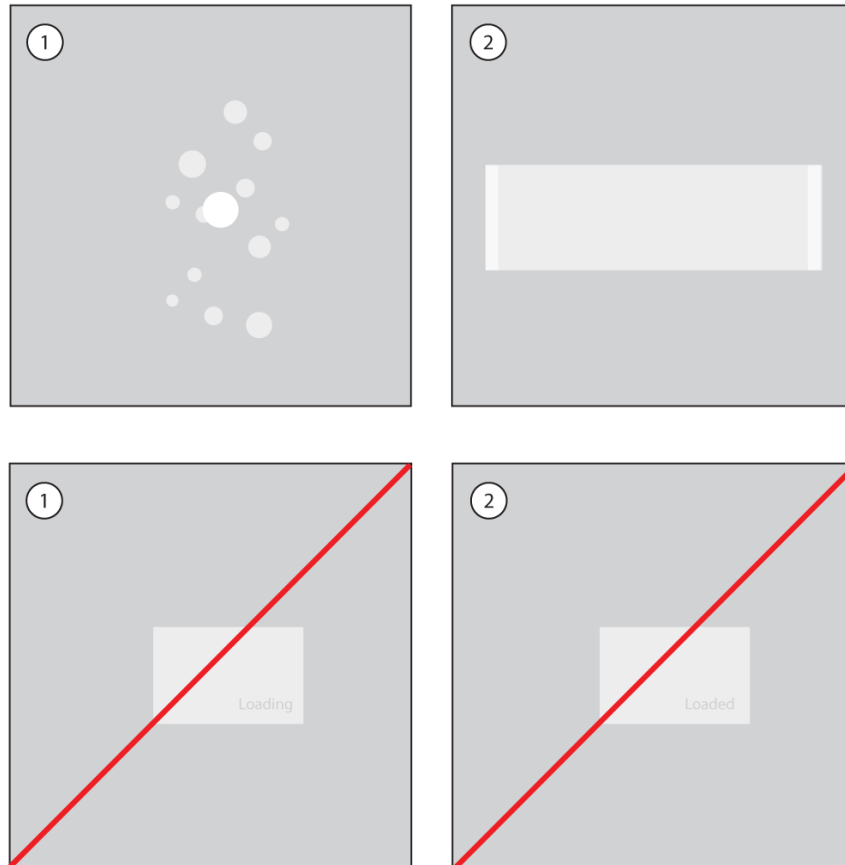


- Understand how people share vertical space.
 - In vertical settings screen space might be sectioned off in columns for several people to use at the same time.

c. Understand how people share screen space

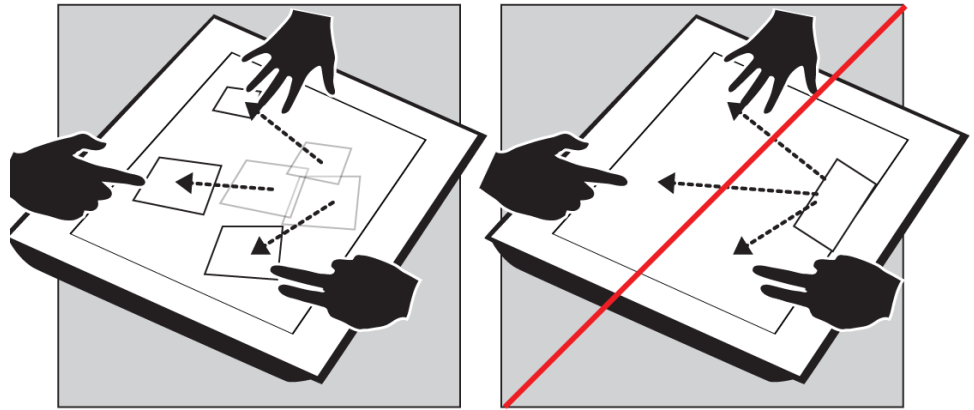
Applications designed for many people to use at the same time need to make it easy for more than one person to see, reach, and touch content.

- Make it clear to everyone when an important application wide change occurs.



- Make sure one person's touch interactions don't unexpectedly affect other people's experience.
 - For example, it's problematic in a paint application if one person selects an eraser control and causes everyone's touches to unexpectedly start erasing content.

- Enable people to move and share controls; do not attach shared controls to one side of the display.



- Communicate content possession through placement; if new content belongs to a particular person, place it in front of them on the screen. If the group shares a piece of content, place it in the center.



d. Support different levels of collaboration

There are three distinct levels of collaboration:

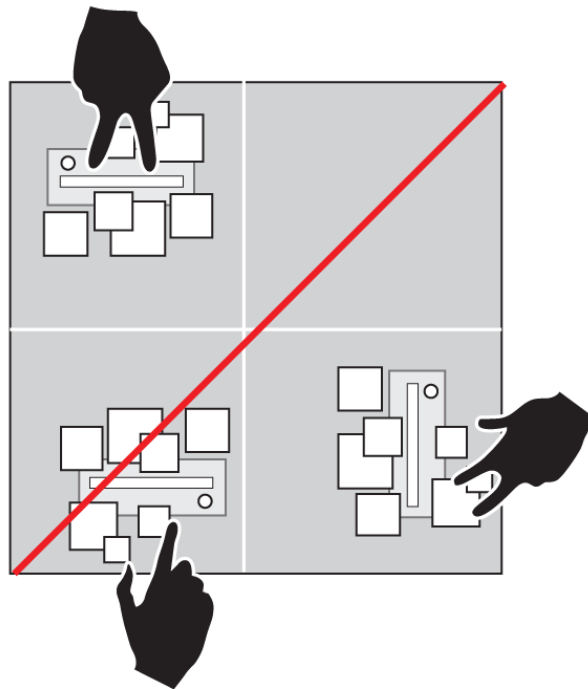
Highly collaborative: People help each other with the same task to accomplish a shared goal.

Somewhat collaborative: People work on separate tasks in order to accomplish a shared goal; also known as "divide and conquer."

Non-collaborative: People engage in separate tasks to accomplish different goals.

Understand how to best support different levels of collaboration within your application.

- Enable several people to simultaneously use content and controls.
 - Do not block progress by requiring everyone to use a common set of controls; allow people to break up portions of the task by physically dividing up and sharing the controls.
- Do not segment the screen into areas tied to particular functions.

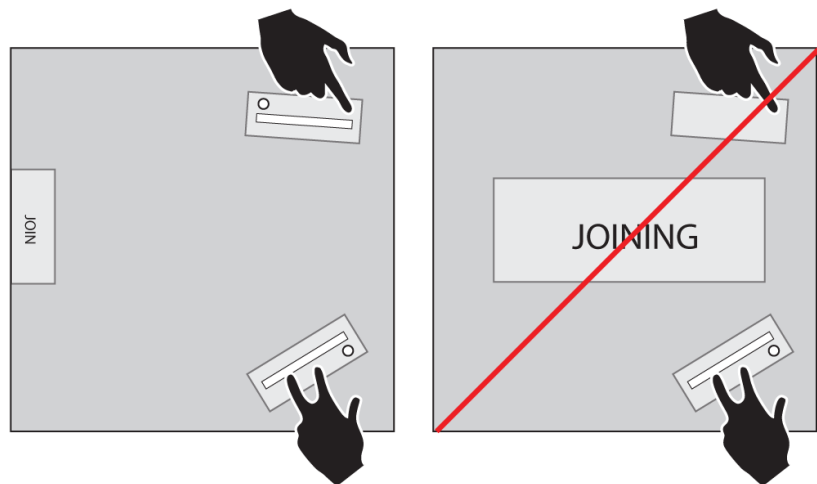


- It is problematic when applications designate one side of the display to perform a specific task, and the other side of the screen to perform another.

e. Understand how people share Surface applications

Surface is optimized for many people to use at the same time, but applications should also be designed for use by a single person. Individuals can use applications and encourage others to join them to share the experience.

- Ensure the number of people using an application doesn't affect its functionality.
 - Enable a single person to enjoy the experience without requiring other people to participate.
 - Enable new people to join without disrupting the experience for people already using the application.



- Make sure your application can continue with fewer people, allowing a person to leave without disrupting all others' experience.
- Create experiences that come alive with several people so that the experience is more fun or efficient when people share simultaneously.
- If appropriate, your application might enable people to divide up their tasks and decide for themselves whether they will be engaged in a collaborative experience or just share the screen to achieve separate goals.

2. Use direct touch interactions

In contrast to the graphical user interface (GUI) mouse and keyboard type of interaction, Surface uses touch and gesture interactions to manipulate onscreen content. Simple

direct touch interactions helps people immediately and intuitively understand the result of their actions, better connecting them to the experience. Using hardware or software buttons to indirectly affect content doesn't typically give them the same sense of control.

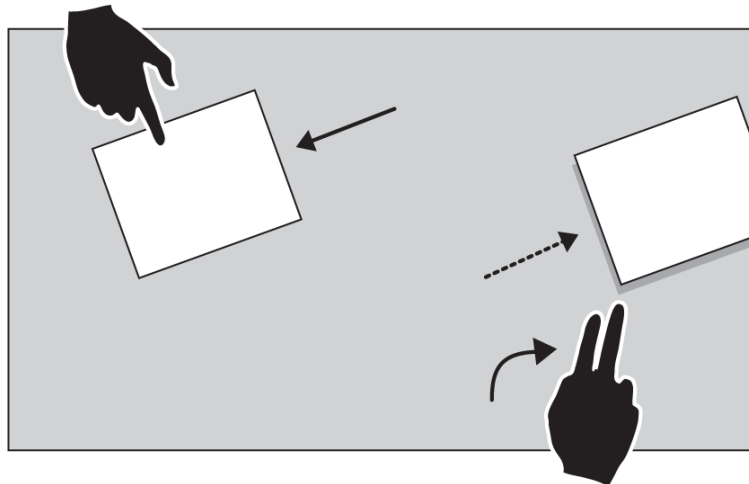
a. Direct touch interactions and indirect touch interactions

Direct touch interactions are physical movements of virtual content within the application by a finger or physical object. Indirect touch interactions usually rely on application interface chrome or abstract gestures. Examples of indirect touch interactions can include buttons, sliders, menus, and gesturing with symbol drawing. Direct touch interactions are the preferred type of interaction for use in Surface because they help to create more intuitive, content oriented experiences.

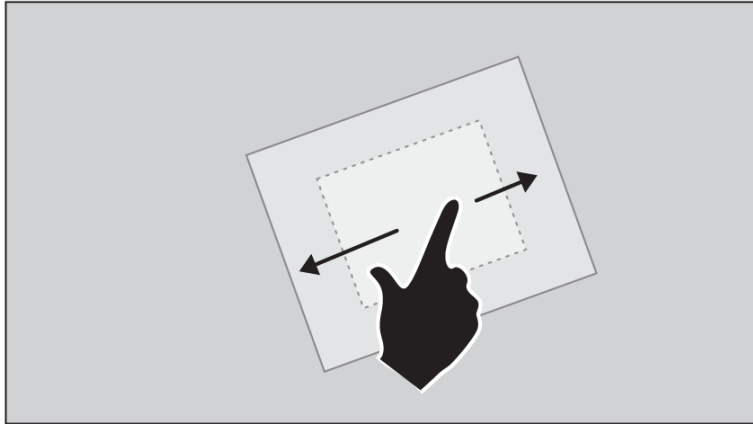
- Do not redefine standard direct touch interactions.

Use the manipulation processor in the Surface SDK to always yield the correct results.

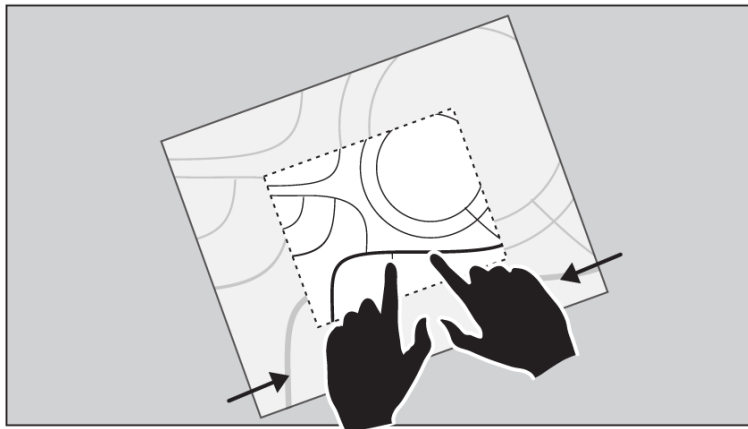
- Move: One or more fingers on an item to move or flick it.



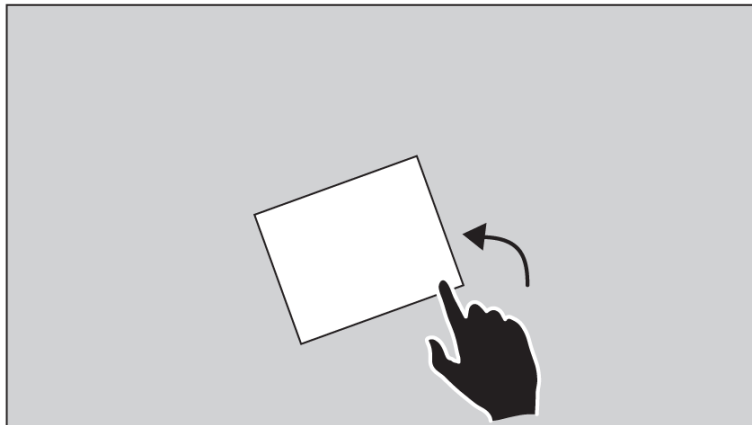
- Resize out or enlarge: Two or more fingers on an item are dragged apart.



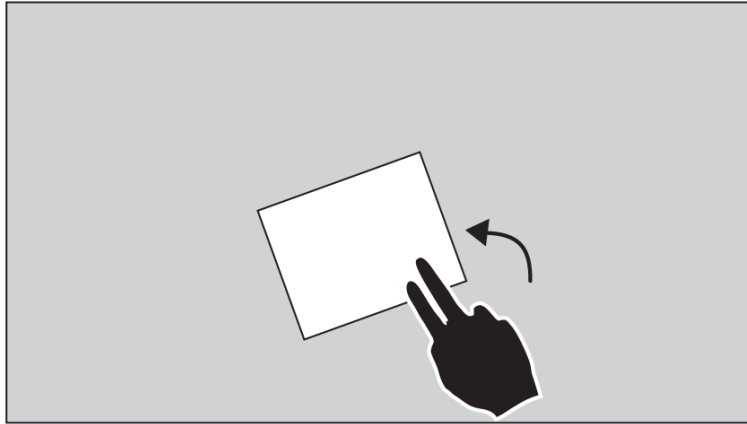
- Resize in or reduce: Two or more fingers on an item are dragged closer together.



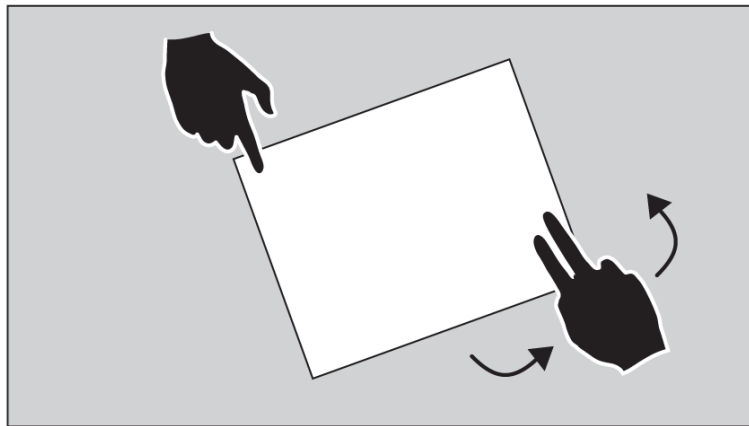
- Single finger rotate: One finger touches an item and drags it around in a circle so that it rotates.



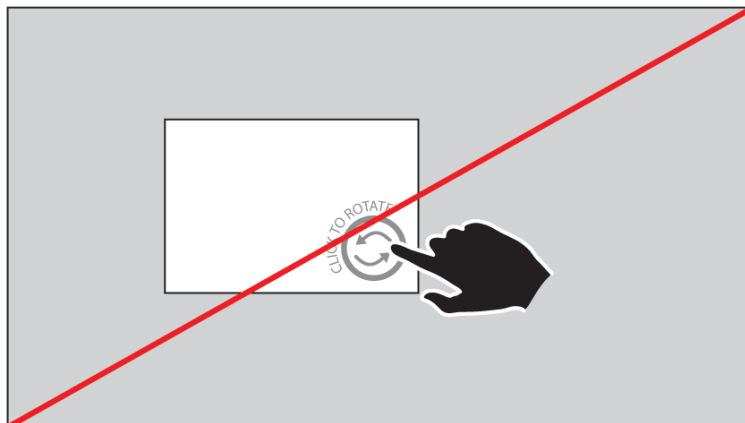
- Two-finger rotate: Two or more fingers on an item are dragged in opposite directions along an arc.



- Pin turn: One finger remains stationary acting as a pivot point while other fingers move around it.



- Don't use chrome or controls to replace direct touch interactions.
 - For example, do not use a rotate button in place of directly rotating an item with a finger.

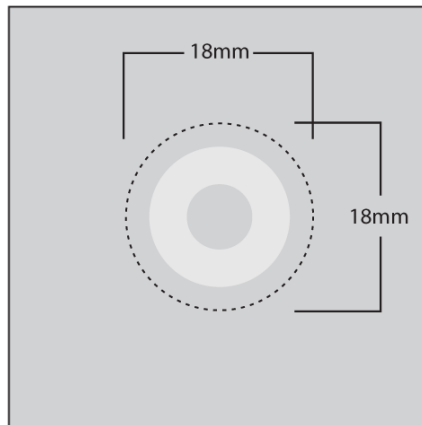


b. Enable single finger drag and flick

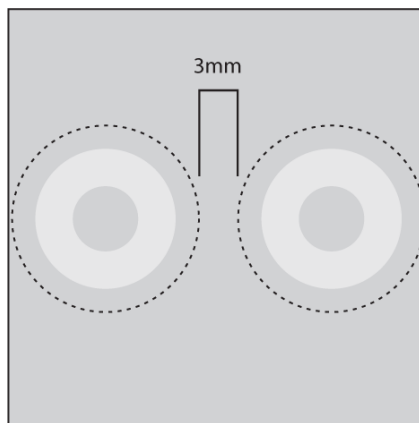
- Enable single finger drag and flick interactions on all movable content.
- Using a consistent single finger drag and flick makes certain that people can always use basic direct touch interactions with all content.

c. Design applications for touch

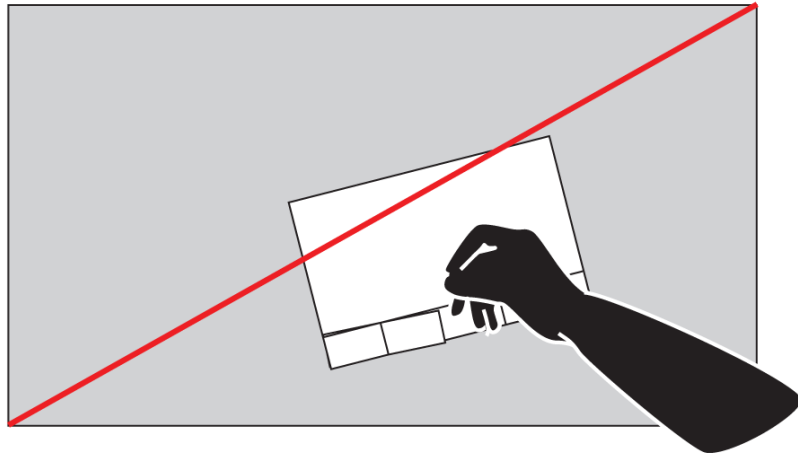
- Interactive elements must be properly sized for finger and object touches.
 - Any item that responds to people's touch, a touch target, must be at least 18mm in size in all directions (18mm x 18mm).



- Allow at least 3mm between minimally sized touch targets.



- Position interactive elements so that hands, arms, and objects covering the screen do not block relevant content surrounding interactive elements.



d. Respond to multitouch

- Microsoft Surface recognizes and responds to over 50 different touches at the same time.
- Your application must respond to multitouch hardware capabilities.
- People expect to use multitouch interactions throughout the experience, not only in self-contained portions of an application.

3. Use physical objects to enhance the experience

People will immerse themselves in Surface experiences, ensure this remains uninterrupted by merging the physical and virtual worlds.

a. Using tagged objects

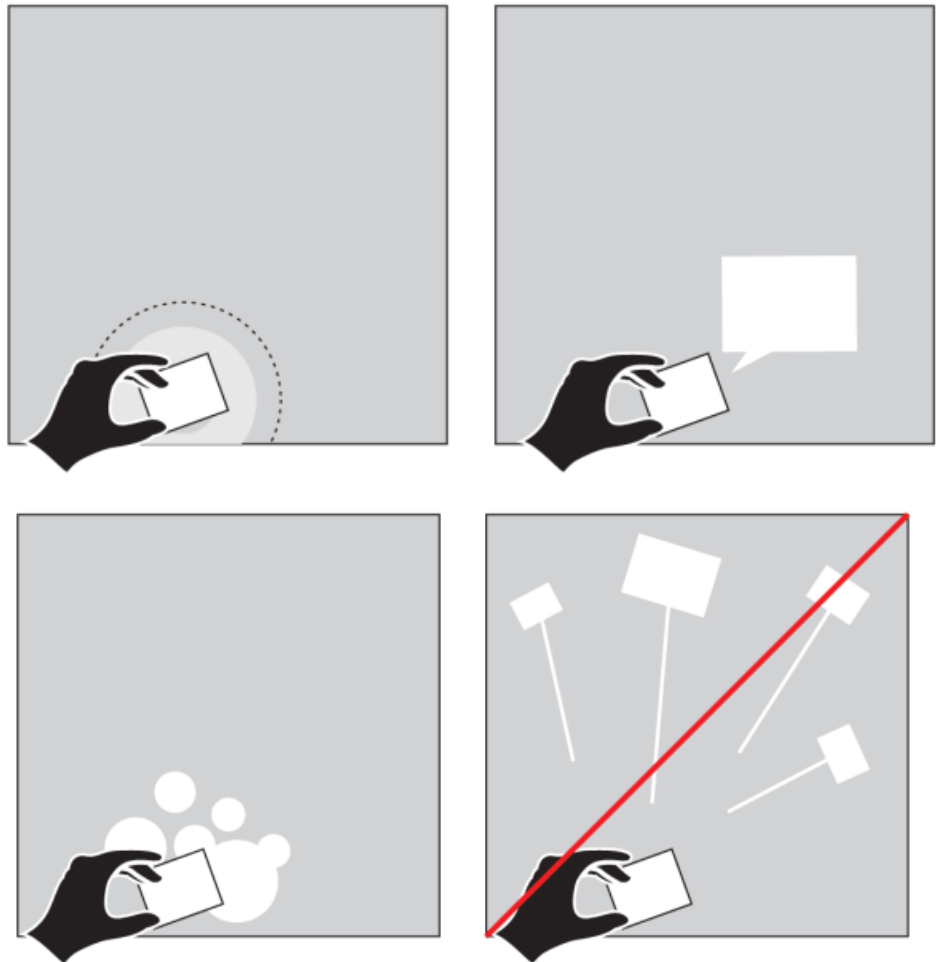
A key differentiator for Surface is its ability to recognize physical objects. Objects help to connect people to the application experience in a tangible way that is not possible with traditional user interfaces.

Placing special tags on an object can help the system quickly and correctly identify it. A tag is a unique pattern of dots that consists of a geometric arrangement of infrared reflective and absorbing areas. When a tagged object is placed on the screen, the vision system reads the tag and determines its value, location, and orientation.

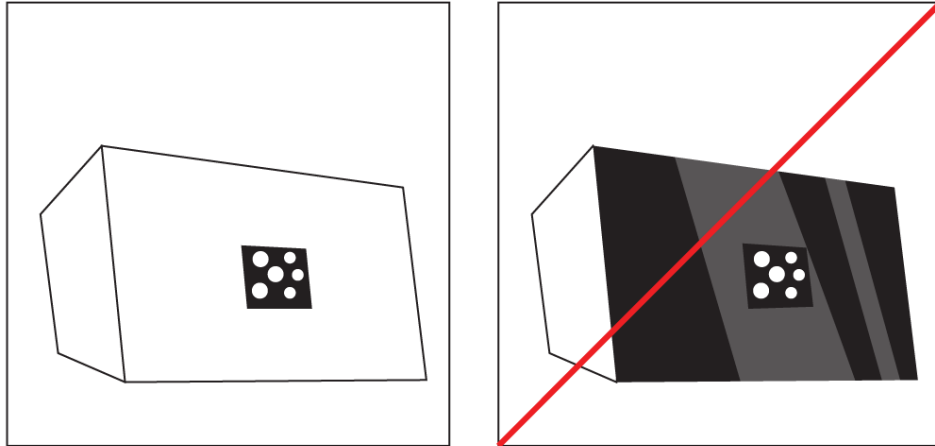
- Use objects to enhance the way people share and own space.
 - For example, an education application might use objects assigned to each student to track lesson progress.
- Use objects to enable playful, delightful experiences.
 - For example, game pieces may have various animations assigned to different pieces. When a player places the game piece on the screen, the tag triggers an animation to play.
- Use objects to create a physical link between people and their content.

Applications that utilize tagged objects should:

- Respond immediately to tagged objects. This immediate response lets people know that the device is working.
- Create a visual response that is appropriate for the object.
- Clearly connect the physical objects with their virtual effects.



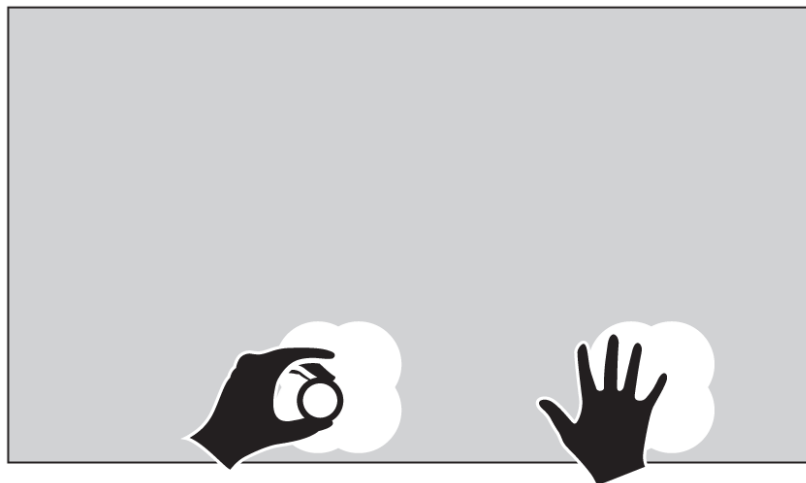
- Use objects to enhance and enrich the experience, but don't require the presence of tagged objects.
- Use non-infrared reflective objects for tagging, so that the reflective portions of the object do not generate contacts within your application.



- If infrared-reflective objects are required, use your knowledge of the tag location and shape of the object to filter (ignore) these contacts.

b. Use untagged objects

An untagged object is referred to as a blob. Surface can detect IR-reflective objects that are placed on the screen. Contacts from untagged objects are the same as contacts from other objects that register as blobs, such as an entire hand placed down on the Surface screen.



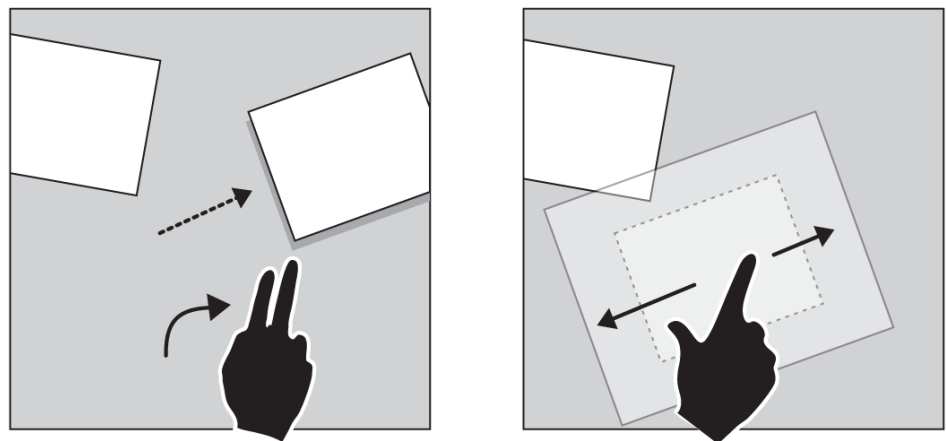
Applications that utilize untagged objects should:

- Respond immediately to the presence of untagged objects. This immediate response lets people know that the device is working.
- Always give a visual indicator when an object is placed on the screen showing that Surface can see it, even if it cannot specifically identify it.
- Do not rely on blob properties to determine the precise shape or size of physical objects. The shape of all objects on Surface is elliptical, and the size denotes only the IR-reflective portions of objects.

c. Make onscreen objects behave like physical objects

Onscreen objects should have subtle physics in performing motions, inertia of movement, and natural feeling collisions to help create a sense of consistency.

- Use interaction metaphors that start with physical manipulation then extend it beyond what is possible in the real world embracing the possibilities of the digital world.



- For example, in a photos application you may be able to move photos around the screen with a flick (mimicking real world interactions) but you can zoom into the photo with a simple manipulation (only possible in the virtual world).
- Mimic the real world in your transitions by using notions such as mass, acceleration, friction, viscosity, and gravity.
 - The ScatterView control in the SDK enables you to easily create these effects.

- Use inertia on objects and content.

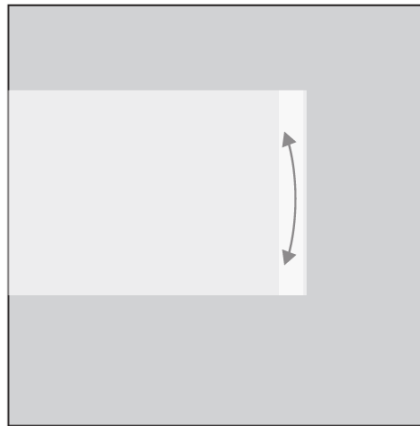
This contributes to creating a consistent experience across applications. The inertia processor in the Surface SDK makes inertia simple to implement. The processor includes two types of inertia: realistic and goal-oriented.

4. Always respond to touch

People might try something that does not work, and the resulting visual feedback should help them learn, resolve problems or encourage them in a positive direction.

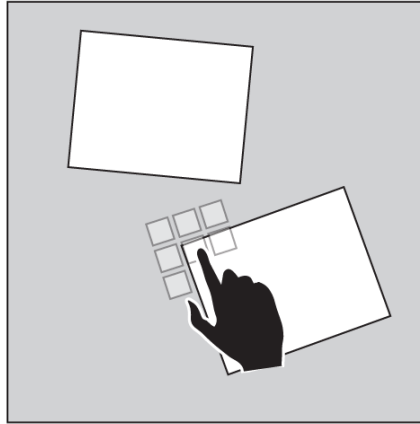
a. Guide people with visual hints and just-in-time chrome

- Give people subtle visual hints as to what the interaction should be.
 - For example, the Launcher rotate handles use a subtle visual hint showing the type of touch interaction that should be used to rotate the control.



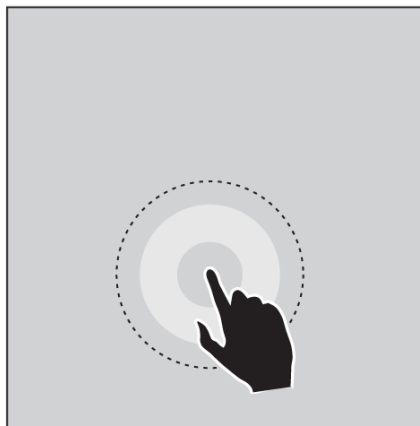
- Use just-in-time chrome to guide interaction. Just-in-time chrome means that controls reveal themselves only when needed.

- For example, in a photos application, ElementMenu should appear upon touching an individual photo rather than being onscreen at all times.



b. Ensure instant gratification and a sense of success

- Visually acknowledge touch.
Touch visualizations quickly and successfully allow your application to visually acknowledge touch.



- All content must respond to touch and must do so immediately and visually. People need feedback that they have successfully touched an object.
- Touch can be acknowledged with depth, glows, or other visualizations.
- Applications must immediately respond to touch input.

- Let people control the experience and do not provide too many automated actions.

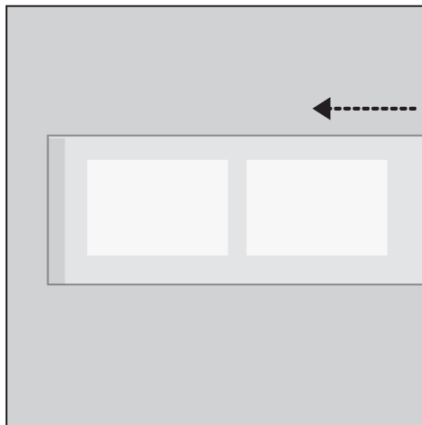
c. Provide engaging sensory feedback

Give feedback that engages people's senses through visuals, sound, motion, and physical interactions.

- Include visuals that are pleasing and appropriate for the scenario.
- Include sounds that are appropriate to the application that help people understand that something has happened.
- Include visual cues and hints that lead people to discover new places, things, or effects within the application.
 - For example, an application can use animation to teach interactions or hint at functionality.

d. Use visual hints and guidance animations to teach touch interactions

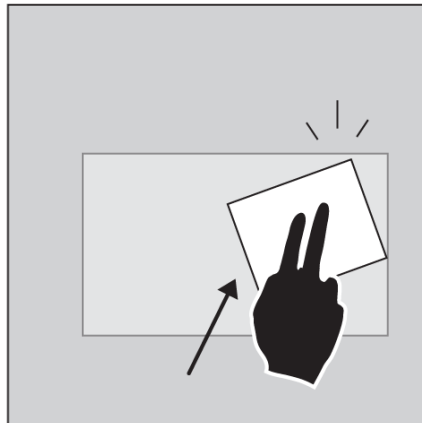
- Provide visual hints for unseen content.



Intuitive, non-menu driven exploration is fundamental to the Surface experience, and sometimes people require clues and hints to guide them in exploration.

- For example, the Launcher animation shows application icons scrolling in and then the icons compress to indicate that the Launcher is scrollable and that there is additional content.

- Provide visual hints that prevent people from doing “wrong” actions.



- For example, do not allow people to move content outside the bounds of a display where they cannot retrieve it.
- Use guidance animations to reveal functionality.
 - For example, in a music browsing application, album covers are ScatterView Items, so that people can touch them and flip them over to reveal the contents of the album. When the application is launched, the covers animate into place and a few albums flip over to demonstrate that additional content exists.

5. Create experiences that are built on content

Content oriented applications encourage exploration and discovery. Experiences that are content focused are more direct, intuitive and enjoyable for people to use.

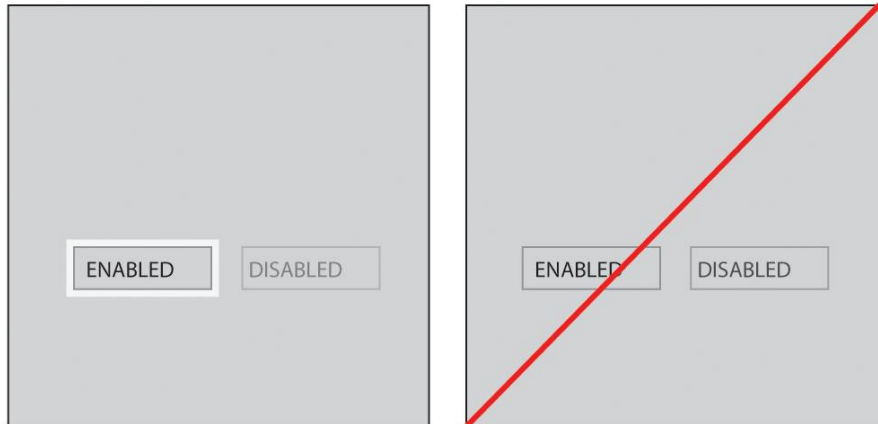
a. Make content the interface

- Do not replace direct touch interactions with UI controls such as buttons, menus and sliders.
- Do not clutter the screen with unnecessary and redundant UI controls and chrome.
- Do not replace touch as the primary interaction.
 - Do not connect external input devices to a Surface unit in such a way that they replace the primary interaction experience.

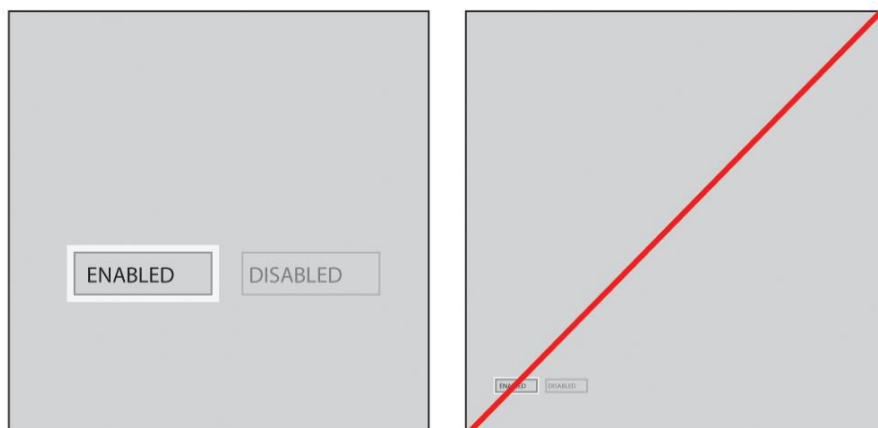
- Applications must support and promote the Surface touch experience, not replace it.

b. Make interactive elements easy to identify

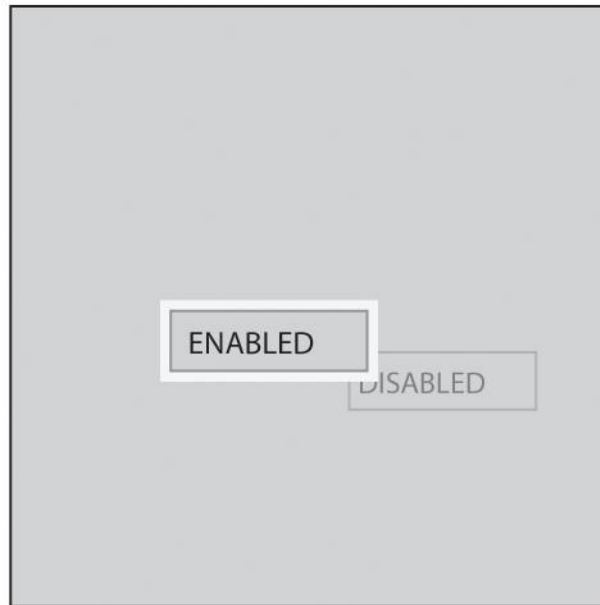
- Make sure visual indicators of touch content are accurate so that people are not misled to touch something that isn't interactive



- For example, a disabled button must be made visually distinct from enabled buttons.
- The larger the interface elements, the more quickly a person can identify it as interactive.



- Use depth to help indicate whether or not an object is interactive.



c. Ensure the experience is focused

- Reduce the number of features in your application so that they account for the needs of the people using the application in a particular scenario.
- Make sure the set of features is focused on particular tasks and that those tasks are clear to people.
- Limit similar choices to reduce complexity and the time needed for people to make decisions.
 - Duplicate controls and interface elements that appear to offer the same functionality are disruptive.
 - When you need to provide a large number of choices, use the ElementMenu.
 - Simple, organized, and hierarchical structure.
 - People can tap or slide to navigate through the menu system.

d. Integrate learning with doing

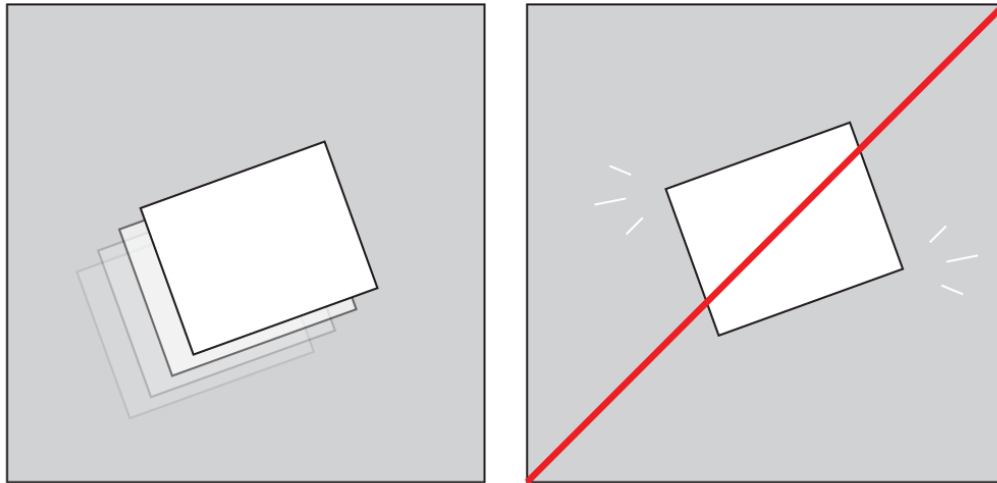
- If interaction instructions are necessary, integrate them with the flow of the application, do not divert focus away from content.

- Make essential features immediately discoverable.

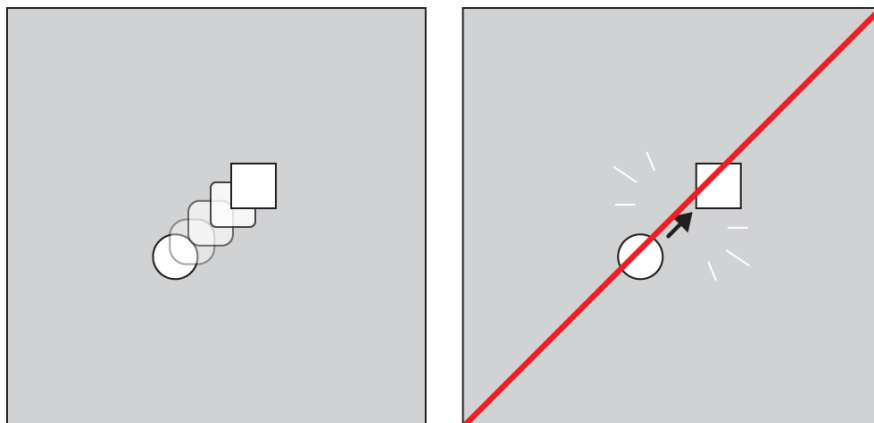
6. A note on transitions

a. Transitions must be fluid and smooth

Smooth transitions give the user context about where they are in the experience.

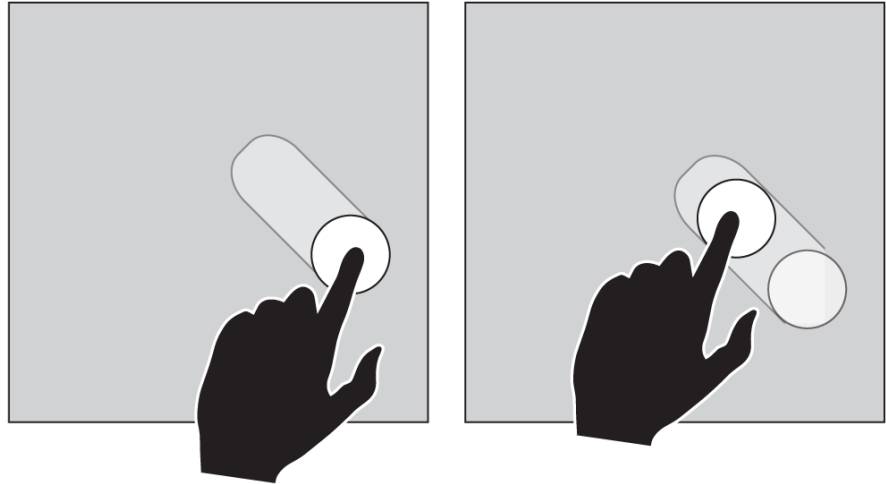


- Make every transition fluid to keep people in a continuous flow.
- Avoid jarring transitions.
 - Smooth transitions maintain continuity and help people stay oriented.
- Nothing should abruptly appear or disappear.
- Every object and visible property change must smoothly animate and transition into and out of existence, or between changes.



b. Avoid unintentional or destructive major transitions

- Require destructive and large changes to be explicitly and intentionally activated by people.



- For example, the Access Points require intentional input to leave an application or start a new experience. This two-step system makes accidental activation of the Access Points unlikely.
- Create a welcoming and friendly application environment that invites people to safely explore and experiment.

Section 3.2:

VISUAL & MOTION DESIGN GUIDELINES

This section provides practical design guidelines regarding visual design and motion design.

1. Orientation & Layout

Laying out visual elements on Surface provides interesting opportunities with the 360° degree nature of the multiuser interface. In most Surface applications, any person from any side of the display should at all times be able to read, understand, and interact with any object on the screen.

There is often no “absolute direction” within Surface applications; there is no explicit top, bottom, left, or right. Instead, it is best to think in terms of position relative to each

person, where an object might simply be rotated or placed towards or away from the people using the device.

a. Application orientation

When an application is launched, ensure that its default orientation is beneficial to most people around the device.

In a *horizontal* deployment:

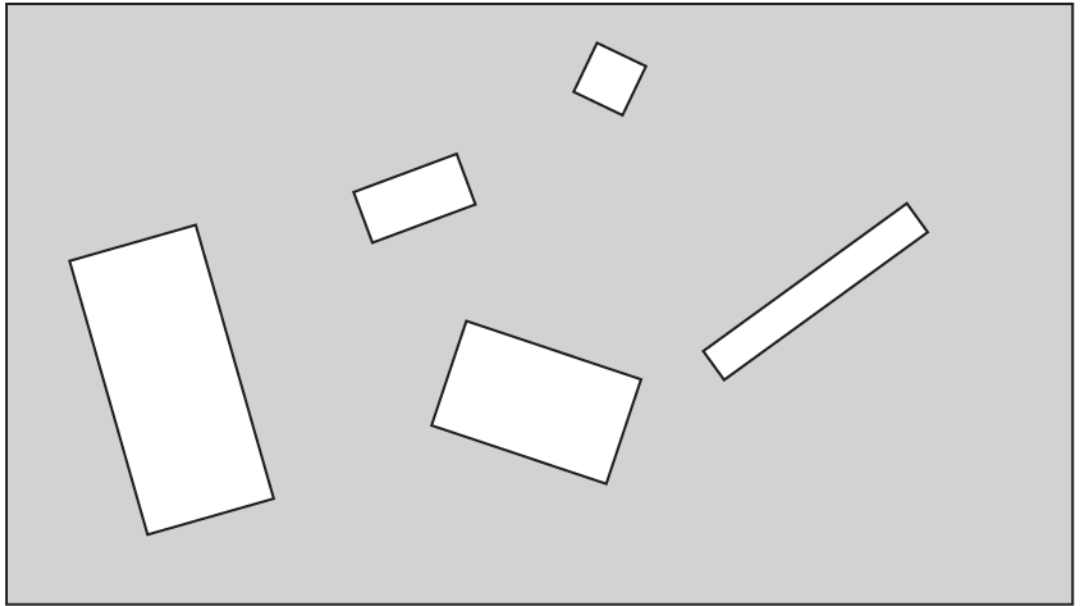
- If an application can have its content freely oriented, that content should be oriented in a way that indicates to people that the content is rotatable and movable.
- If an application must be facing in one particular direction, it should follow the same orientation as the Launcher.

In a *vertical* deployment:

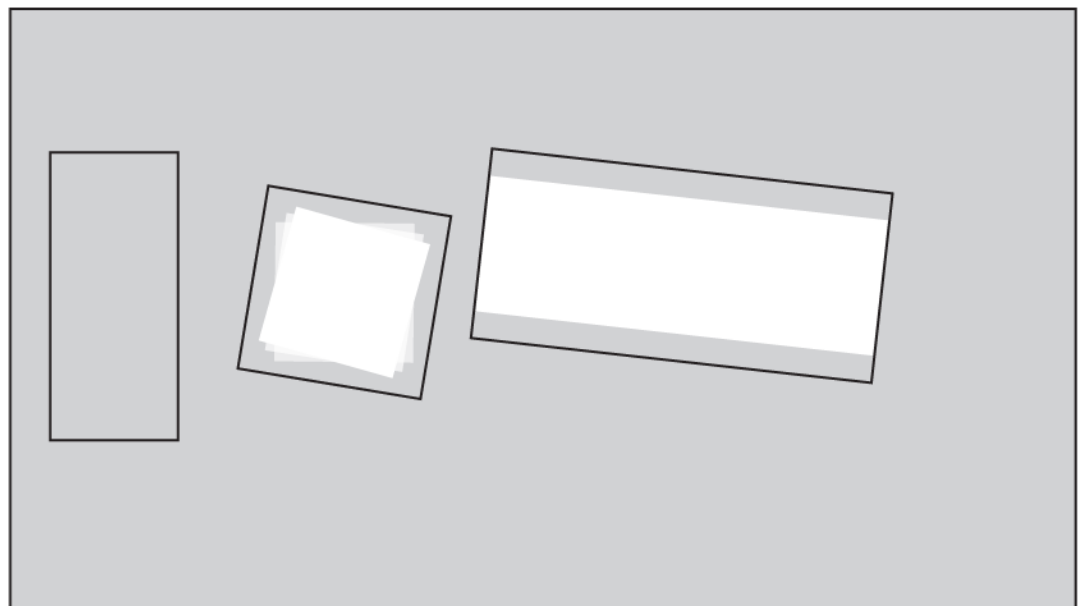
- The Launcher can only be oriented towards the bottom of the screen and applications should follow the same orientation as the Launcher.

b. Grid-free layouts

Most visual designers have learned to create layouts based on grids. The 360° degree nature of Surface is great for laying out applications without a global, or screen wide, grid system. This requires a fresh perspective on visual layout; the Surface SDK ScatterView control is a quick and easy way to create grid-free global layouts. It encourages people to organize and explore content. ScatterView acts as an invisible container for onscreen objects, enabling some content to be oriented towards each edge of the screen by default, which encourages curiosity, direct touch interactions, and exploration.



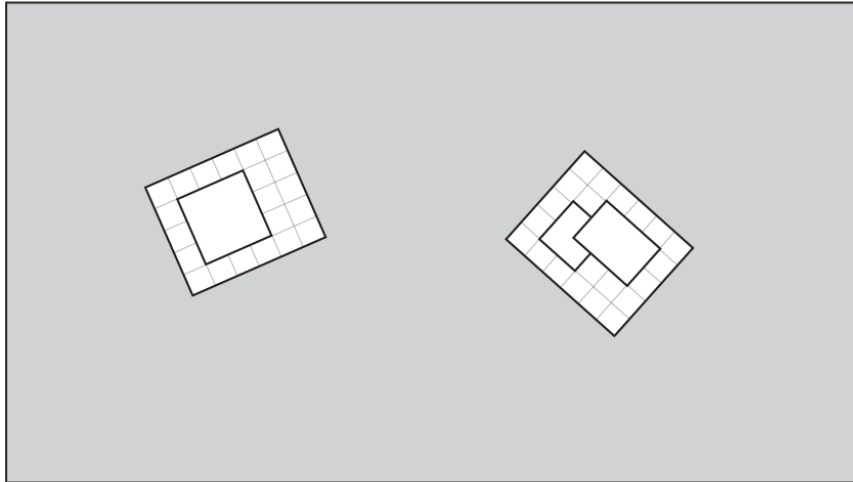
Layouts don't need to be grid-free all the time. For example, the `LibraryContainer` control is a quick way to create order in a local section of a grid-free layout so that content may be easily sorted, filtered, and organized. Applications can also let people decide when to switch between different visual organization methods.



c. Gridded layouts

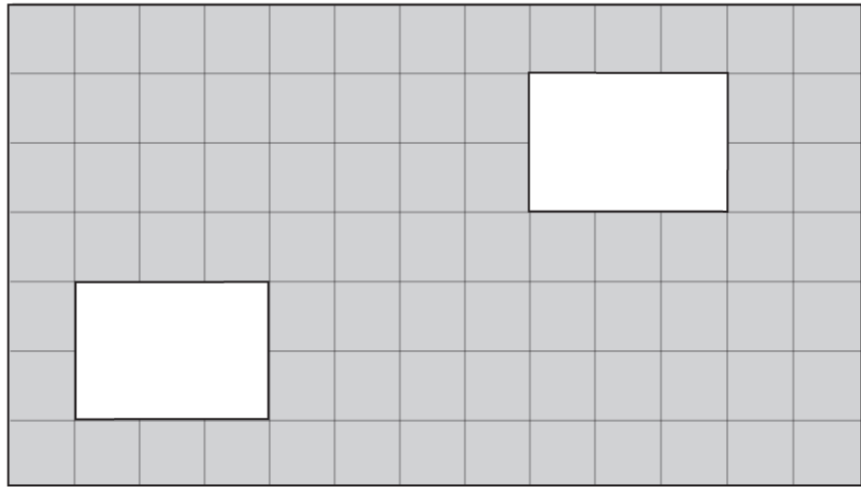
Grid based layouts still have their place and can be useful in certain Surface applications. Gridded layouts are ideal for productivity or focused activities, linear sorting of data, or simply to create a visual rhythm. Layout grids for Surface experiences can be local (specific to a particular content item) or global (screen-wide).

- Using local grids



These are layout systems that apply to specific objects, not to the screen as a whole. A particularly useful implementation of local grids is within a ScatterView item; your application can enable 360° degree experiences with ScatterView, but the local grid can be used within ScatterView to give a feeling of organization and structure.

- Using global grids



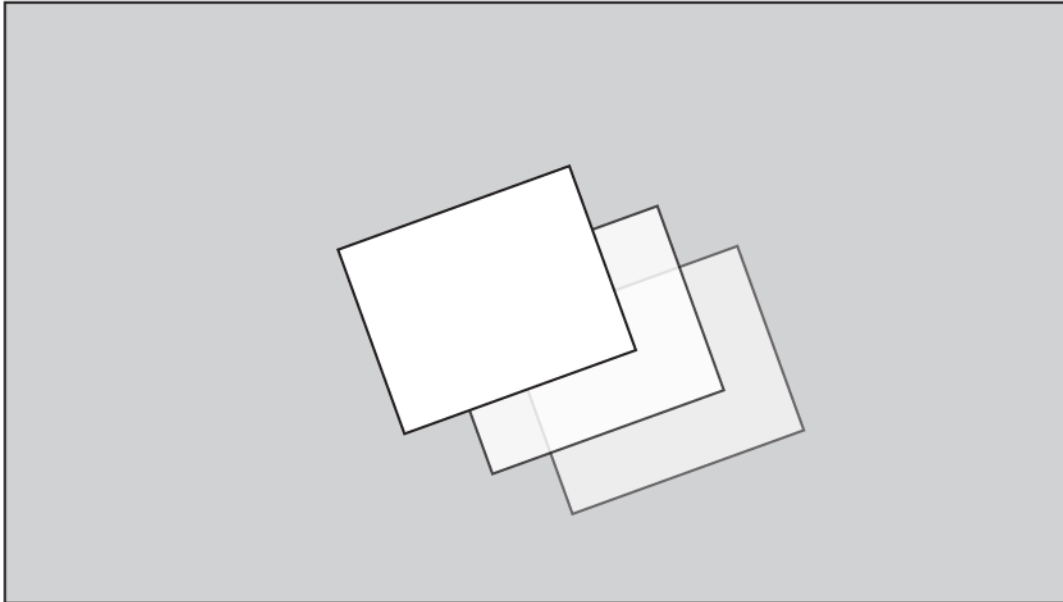
Global grids determine the arrangement of content and controls for the entire application. Screen wide global grids are useful when duplicate controls are offered to each person, or to each side of the Surface screen.

While permitting easy scanning and organization, they often force the entire UI to be oriented towards one side of the screen and make it difficult for people to collaborate. If you are laying out elements on a global grid, give careful consideration to content orientation for every person on each side.

2. Depth

Using the z-axis, or depth, for content organization and visual prioritization can be very powerful. Depth helps separate foreground elements from the background and can also provide clear separation between interactive elements and non-interactive elements.

The Surface interface uses depth throughout the experience and in transitions between various states such as Attract, Launcher, and applications.

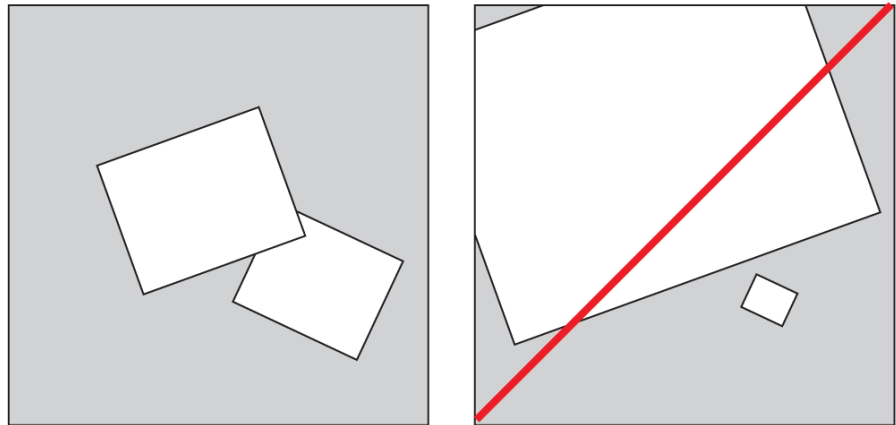


- Use depth to show priority, order, or focus among content items.
- Keep apparent depth shallow, subtle, and elegant.
- Use depth to acknowledge successfully touched objects and controls. Give the illusion of content floating toward a finger to register a successful touch, reserving the highest level of depth for those objects actively being touched.
- Use depth within content and controls to visually indicate touchable regions.

a. Creating depth using 2.5D

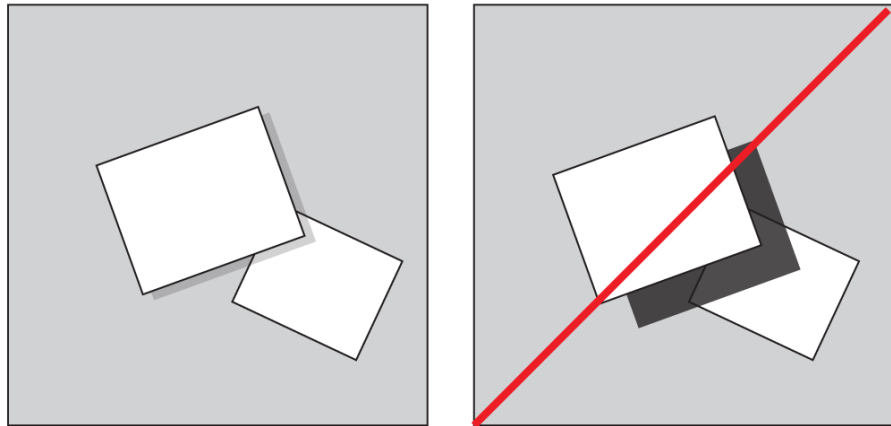
2.5D is typified by using techniques that simulate or create the illusion of depth without actually requiring 3D geometries. The various techniques can be done without specialized 3D tools using XAML and the Windows Presentation Foundation (WPF) in a high performance way.

i. Scale



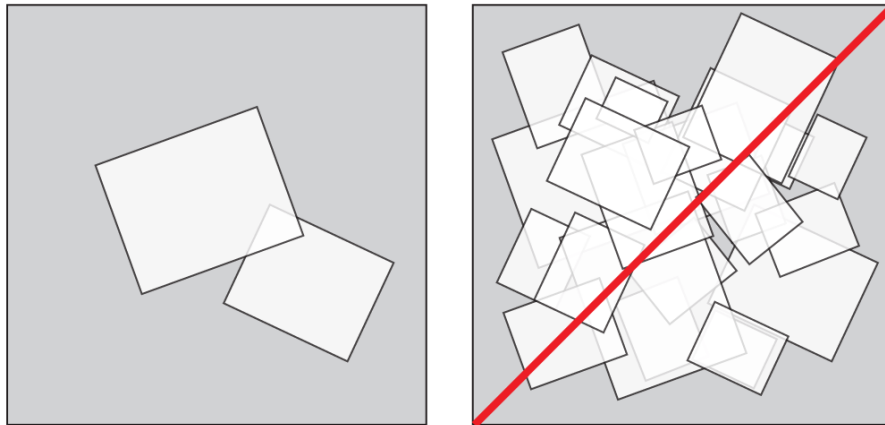
Scale can be used to simulate depth, especially when combined with other techniques. Scaling is an ideal way to register successful touch input; a slight increase in scale (or the illusion thereof) gives the impression of an object being magnetized to someone's finger. Large differences in scale between different levels of depth can lead to an appearance of an extremely deep background. Use small, subtle changes in scale to keep people immersed in the experience.

ii. Drop shadows



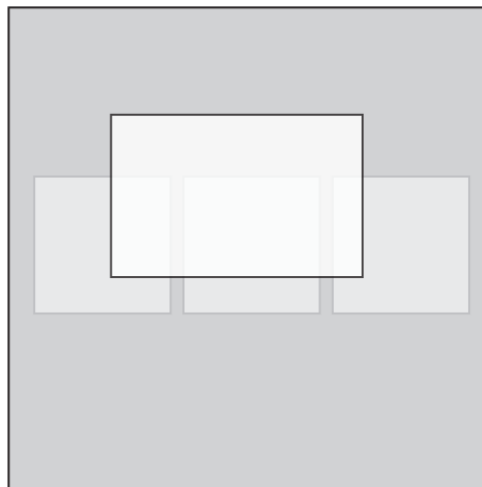
Subtle drop shadows are an easy and effective way to indicate depth, but it is important not to make them too heavy. Dark shadows can obscure content below, and very large offsets can make objects feel disconnected from the Surface display.

iii. Transparency



Transparency can show depth by allowing objects on lower levels to show through objects above them. This technique is good for a small number of objects onscreen at the same time as transparency can be computationally expensive to render.

iv. Depth cueing

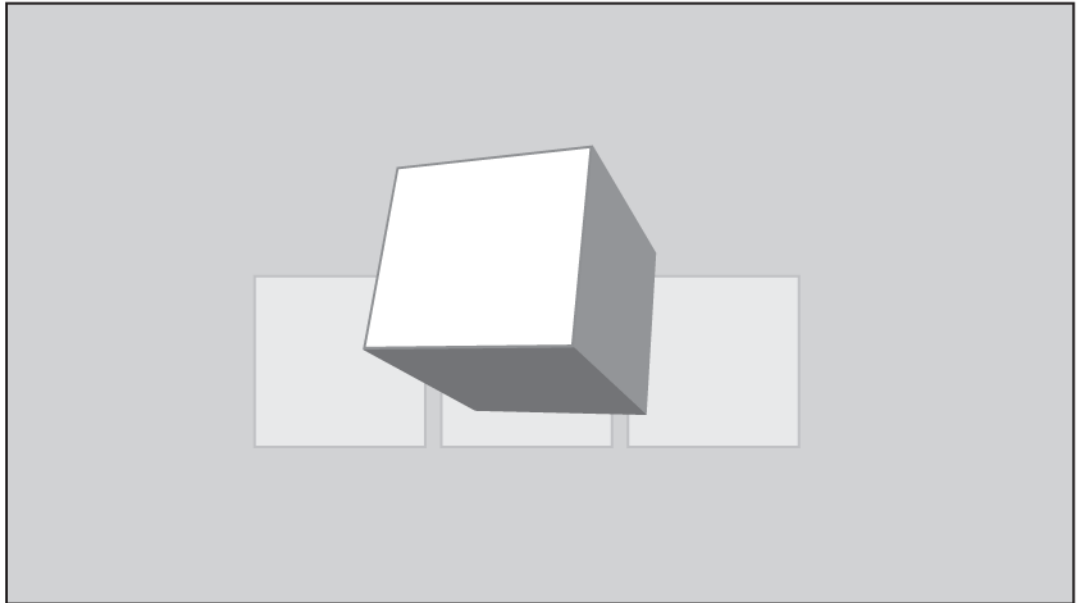


Depth cueing uses tinting, saturation reduction, and/or contrast reduction to simulate distant objects. Subtle reductions in brightness to convey depth and object focus can be quite effective.

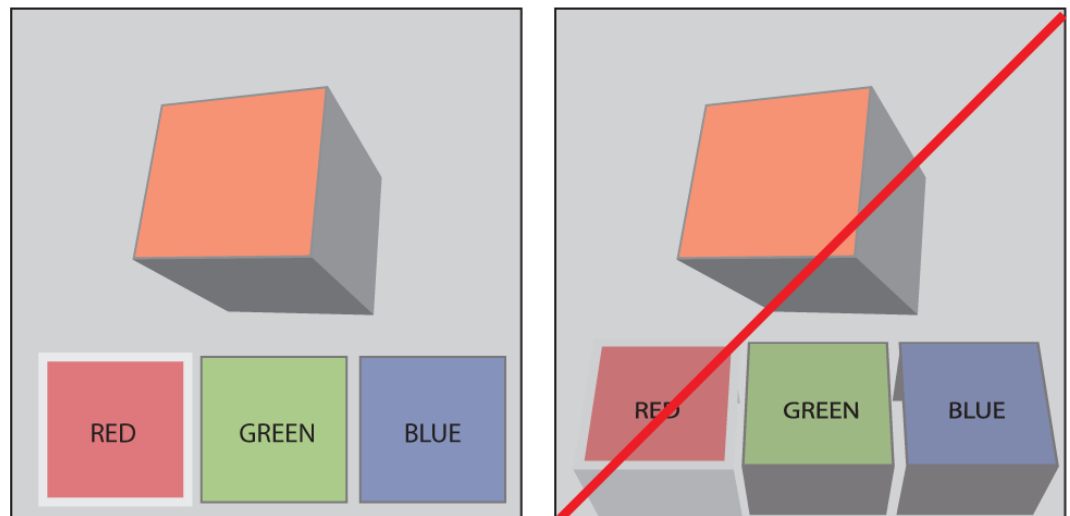
b. Creating depth using 3D

True 3D uses rendered three-dimensional geometries in real time. This enables realistic rotation of cubes, spheres, custom 3D models, and so on. While 3D can

be authored in XAML and delivered using WPF, it can adversely impact application performance. True 3D is best created and delivered on Surface using XNA, the core Microsoft 3D and gaming engine.



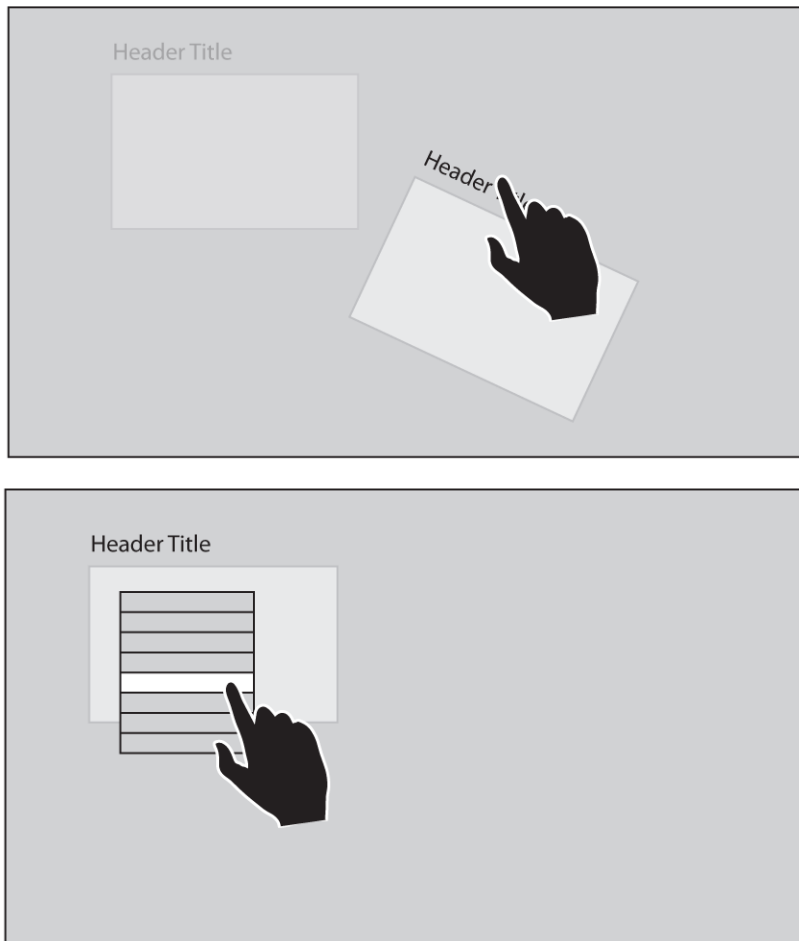
True 3D offers many opportunities for interface innovation, but use caution when designing 3D content. Controlling a 3D object on a two-dimensional touch surface can be difficult and confusing. 3D should usually be avoided for the creation of controls, as any text or labels on 3D surfaces can become difficult to read.



Consider using 3D depth not to describe onscreen objects, but instead as a virtual space to guide people through the application.

c. Depth as visual hint

Depth can also be used to indicate interactive zones for touch input. This helps people understand that touching in one region will have a different result than touching in another. For example, the header or title bar of a content item is likely to be dragged, but a list item within that object will scroll when dragged. Making one portion of a control appear above or beneath other elements within that control can help people identify these zones and make their purpose easier to understand.



3. Shape & Form

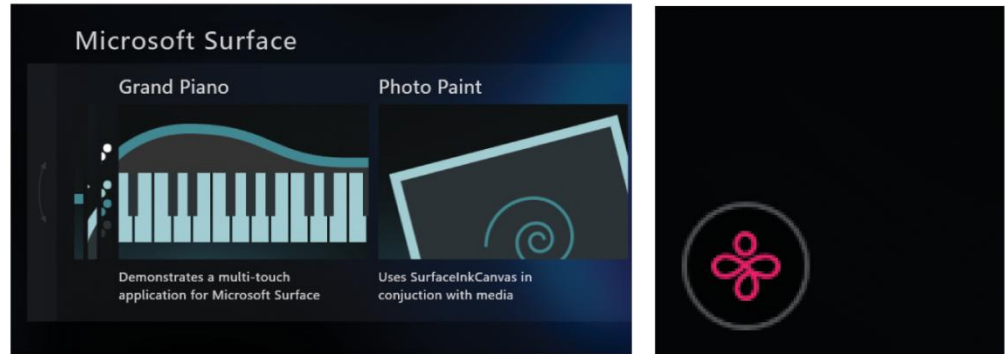
The shape and form of Surface content should visually indicate its function.

a. Edges and corners

The styling of content edges and corners in applications will affect the kind of experience people have.

Looking at the core Surface interface, two styles are used to communicate different messages to people:

Style 1: Flat



The Surface Access Points and Launcher are meant to be clean, flat, simple, and digital. This style indicates that the Access Points and Launcher are interactive elements used to navigate through the Surface experience. These navigation controls are secondary to content. They are not distracting, and they help people get to the next phase of the experience.

Style 2: Organic



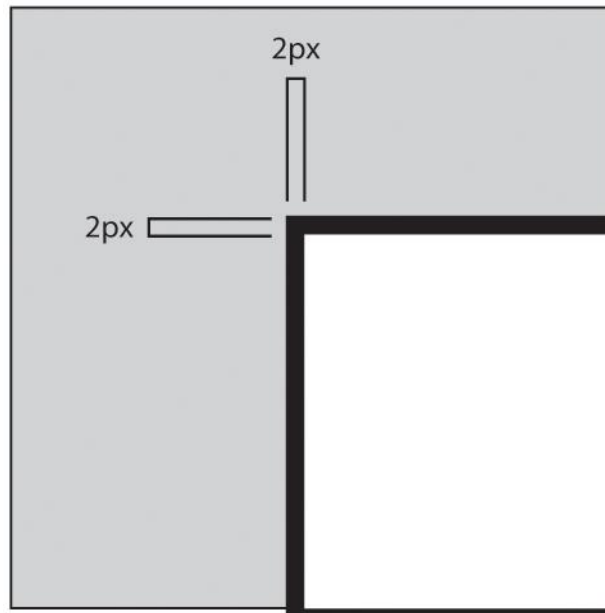
The Surface Attract mode is based on volumetric, organic forms that encourage exploratory touches. Rounded, organic forms are generally preferable for approachability, discoverability, and engagement.

Balancing these styles will help achieve an experience that is appropriate for your application.

b. Line work and outlines

The Surface interface is typically free of outlines, keeping focus on the content.

If an interface element requires outlines, keep line work to a two pixel minimum width to produce best results when rotating or scaling.



4. Texture

Generally, using textures as pure decoration is not recommended for Surface experiences. Ornamental textures tend to detract from content, which should always be at the core of the experience. Textures used for purely aesthetic reasons typically create experiences that are not authentically digital.

There may be certain applications where textures themselves *are* the content. If texture must be used in your application, keep them minimal, consistent, and always stay focused on encouraging touch and exploration.

While textures can act as visual hints to help people use the application, many other visual design techniques can achieve the same goal: negative space, form, shape, color, and more. Explore alternatives to using textures as visual hints.

5. Colors

Designers should follow established color design guidelines, and take into account the specific requirements of the brands they are handling. There are some additional guidelines that optimize color reproduction for the Surface display.

a. Using colors

- Proof all colors on Surface during the design process; iterative proofing is the best way to achieve consistency and optimize brightness and color for the Surface screen. Do not rely on a desktop monitor for proofing.
- Never use pure white as a background color.
- Bright or saturated tones are best in smaller regions to reduce eye fatigue.
- High contrast, such as pure black on pure white, reduces the effectiveness of anti-aliasing (the subtle edge blending of foreground elements onto the background), making text look harsh and introducing a jagged stair-stepping effect into object and content edges when rotated.
- Treat the background as a stage against which all action occurs; it should be subtle, calm, and understated.
- Use subtle gradients cautiously; the reduced color contrast may cause subtle gradients to flatten and appear as one continuous tone.
- Test gradients on the Surface screen to check for banding.
- Experiment, iterate, and explore as you become familiar with the Surface display characteristics. Keep an open mind towards taking fresh approaches to a brand's established color palette.

b. Surface colors

Surface administrators control the default color scheme of the Surface system. This global color scheme determines the default colors of Surface controls that are rendered in all applications. You can use those same colors to style additional elements in your application. However, you can also override the Surface colors to affect any element in your application that uses those colors.

Surface comes with the following four pre-defined color palettes, defined in the classes of the `Microsoft.Surface.Presentation.Palettes` namespace:

- `Light palette`
- `Dark palette`
- `LightHighContrast palette`
- `DarkHighContrast palette`

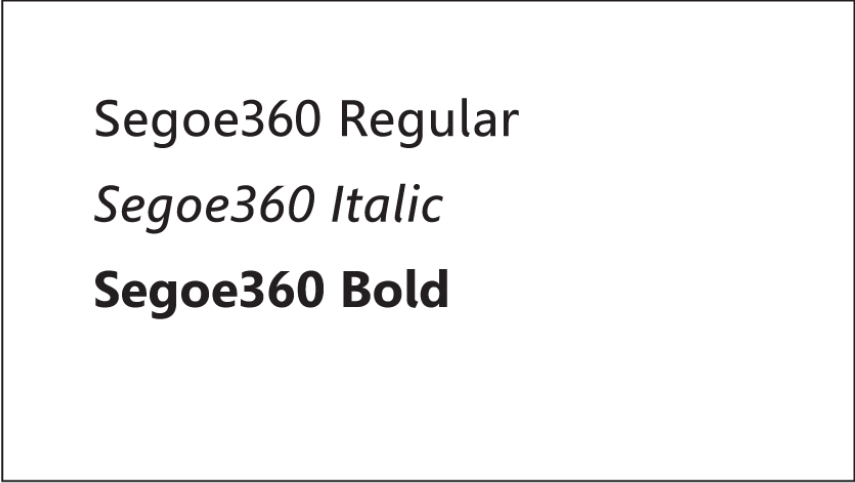
6. Typography

Typography in Surface applications should be elegant and meaningful. With simple interface elements and content at the core of the Surface experience, onscreen text has even more opportunities to act as the interface itself.

a. Segoe360

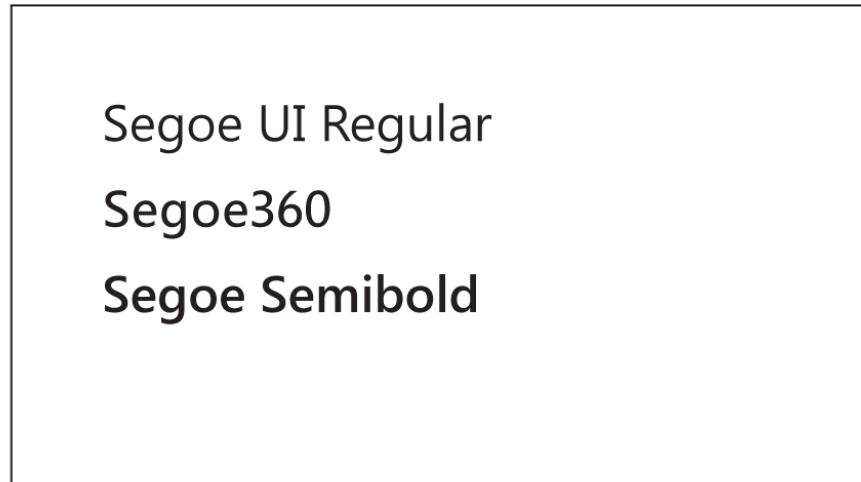
Segoe360 is a specially designed typeface for use on Surface. It's strongly recommended that you use Segoe360 in your applications for the following reasons:

- Segoe360 comes with three weights- Regular, Italic, and Bold.



Segoe360 Regular
Segoe360 Italic
Segoe360 Bold

- Segoe360 Regular is a new weight between Segoe UI Regular and Segoe UI Semibold, designed to have better presence on Surface.



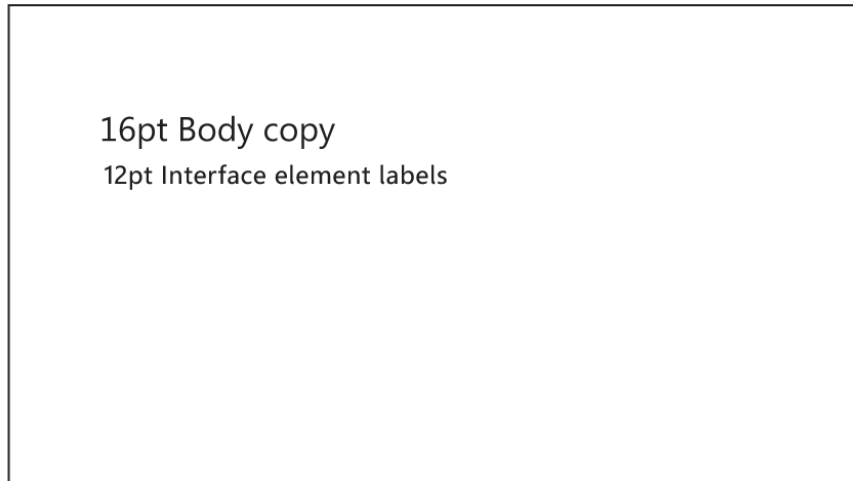
- Characters have been modified for better distinction at all viewing angles, enabling better 360° degree readability. For example, the lowercase “l” is more easily distinguished from a capital “l” and a lowercase “q” is more easily distinguished from a lower case “b” when rotated.



- Tracking (overall letter spacing) has been increased to be more legible and recognizable onscreen, especially at small sizes.
- Counters (the enclosed areas in letters like ‘a’ and ‘e’) have been opened up so they retain their visibility at small sizes.
- Segoe360 supports Western Europe, Eastern Europe, Turkish, Baltic, Greek and Cyrillic character sets.

Minimum Segoe360 type sizes:

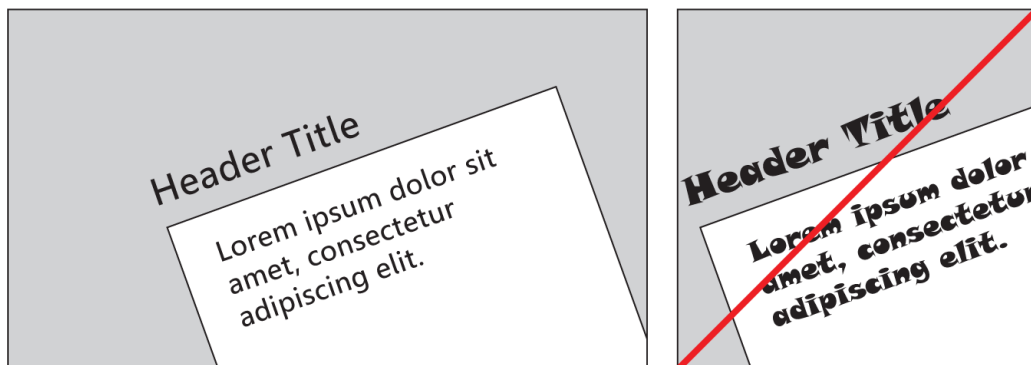
- 12 points for interface element labels
- 16 points for body copy and text



b. Selecting typefaces

Segoe360 is strongly recommended for use in Surface applications, but it may not always fit the needs of a particular brand or scenario.

When choosing an alternative typeface you need to consider rotation, legibility, and orientation. Sans serif typefaces – those without small projecting extensions at the ends of character strokes – are typically preferred over serif typefaces, due to their minimal flourishes, consistent widths, and generous negative space. Take great care when selecting and using alternative typefaces for Surface; be sure to test legibility on a Surface display before finalizing selections.



- Evaluate text rendering at all angles before and during application development.
- Avoid using all capital letters and small capital letters. Sentence case capitalization is more natural, easier to read, and rarely conveys an aggressive tone.
- If serif fonts are an integral part of a visual brand experience, set them at 30 points or larger for maximum legibility.
- Typesetting in high-contrast reduces legibility. White text on a black background can create harsh edges and a poor reading experience.
- Avoid setting text on curved paths.
- Remember that most Surface objects and controls can be scaled freely. Enforcing minimum scaling limits on objects is sometimes necessary to retain maximum text legibility.
- There is no substitute for testing font rendering on Surface itself. Make sure to test for legibility, including when text is rotated and scaled.

Minimum alternative type sizes:

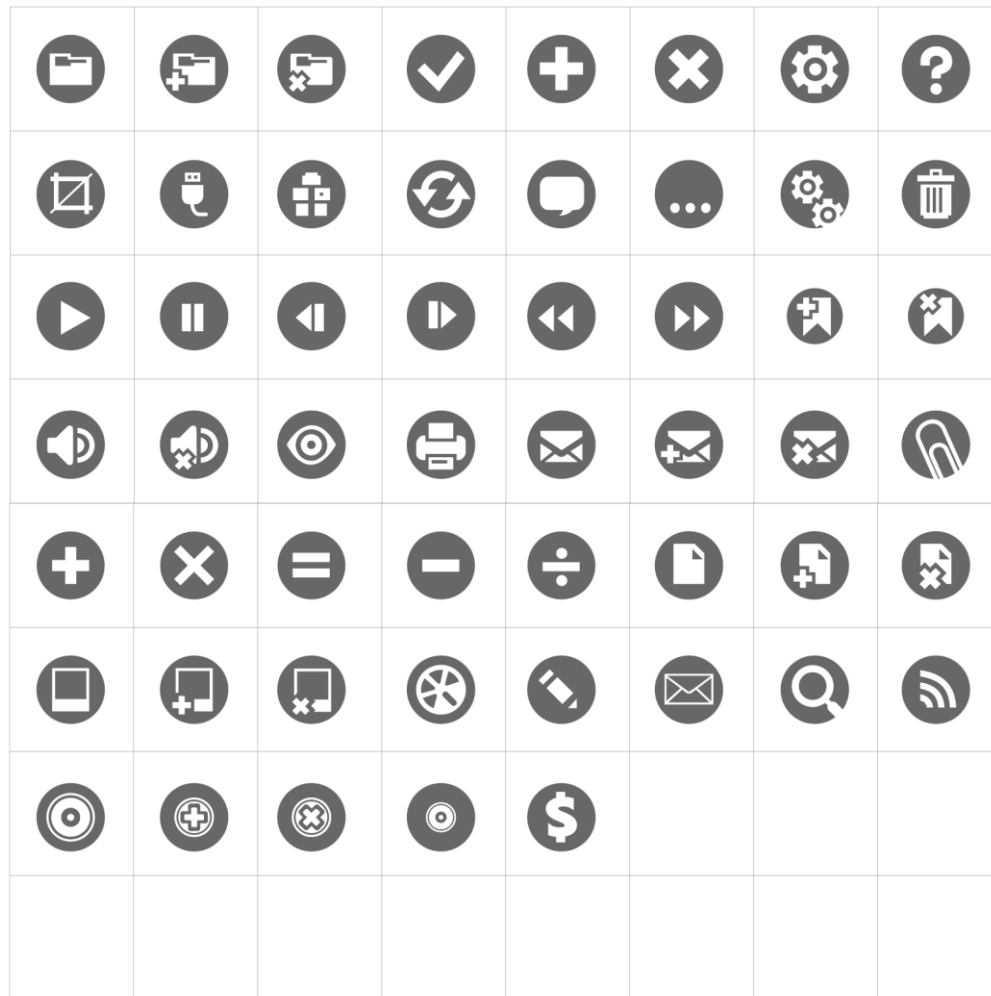
- 12 points for interface element labels
- 16 points for body copy
- 30 points for light typefaces (thin and/or narrow width)
- 30 points for serif typefaces (containing small projecting extensions at the ends of character strokes)

7. Iconography

Icons can help the Surface experience, but usage must be meaningful.

a. Icons as visual hints

Icons are useful to visually convey functionality. The function of buttons and other controls is not always obvious to people, and it is not always feasible to use text labels on small controls. In such cases, simple yet bold icons can be used as a supplement to textual button labels. A set of more than 40 sample icons is distributed with the Microsoft Surface 2.0 SDK.



Do not use icon buttons when suitable direct touch interactions are available, use icons to hint at functionality. For example, the Surface Launcher is rotatable. Rather than placing a button with a rotate icon in the corner of the Launcher, the Launcher uses subtle rotation arrows to indicate the action people should take.

b. Creating custom icons

If an icon is needed for your application that is not included in the sample icon set use the following guidelines to develop custom icons:

- Icons should have simple geometry.
- Icons should have limited detail.
- Icons should be scalable.

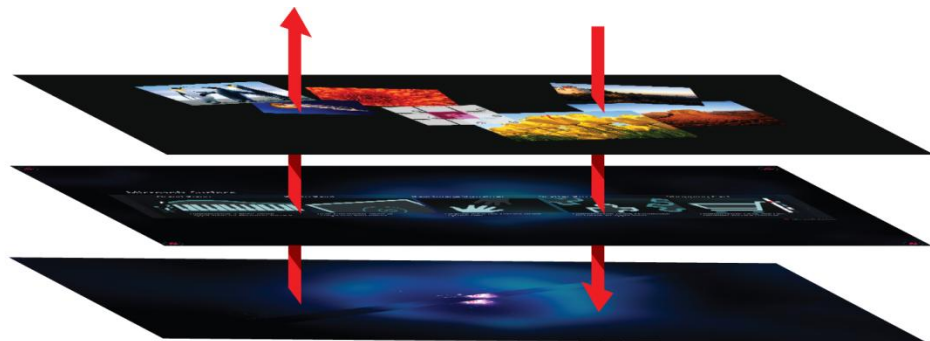
- Icons should leverage real world metaphors.
- Icons should have a similar visual weight.

8. Motion Design

Motion design defines how things move onscreen, a critical part of the Surface experience. Animations always support the content and the experience as a whole. Motion design provides animations and effects that convey emotion, energy, connection, and responsiveness. They provide visual hints, cues, and an invitation to explore content.

a. Transitions

Transitions can give context and provide visual hints of where people are in the Surface experience. They help people build mental maps of their experiences, learn how to use controls, and remember where onscreen objects have gone when no longer visible.



The Surface transition model for shifting between various modes (Attract – Launcher – applications) uses depth to provide navigational context. For example, when you leave Attract to enter Launcher, Attract pushes back in z space – it moves behind Launcher. When you launch an application from the Launcher, the Launcher moves back in z space and pushes Attract back even further. This gives people important context regarding the way they are moving through the experience.

Develop a transition language or model that is appropriate for the function, character, and style of your application. Consider the type of application that you are developing and create transitions that are fitting and appropriate.

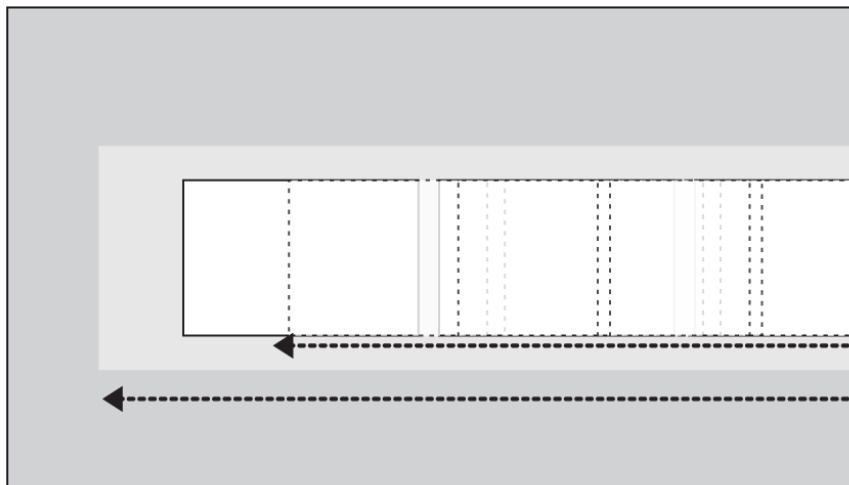
Transitions that occur in a game may be fun and playful, while transitions that occur in a healthcare application should be brief and succinct.

Transitions should:

- Be seamless
- Be meaningful
- Be informative
- Be application appropriate
- Be carefully timed. Don't frustrate people by forcing them to wait through long transitions just to access application content.
- Provide navigational context
- Reinforce the application narrative

b. Transitions as visual hints

Motion design provides tools that enable people to rapidly learn through observation; without requiring directed tutorials or lengthy demonstrations.



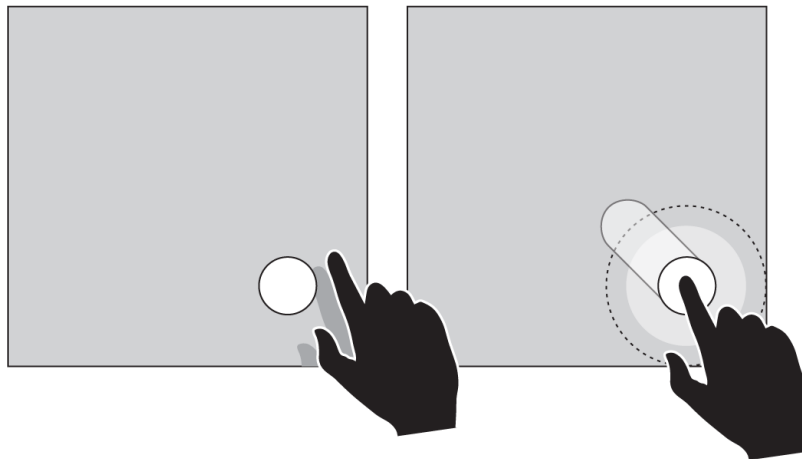
For example, when the Launcher loads a background container fades in, the application icons slide in, and the application icons compress as they reach the edge of the background container. People see this movement and understand that there are more options than are shown. The sliding movement visually hints that the content is scrollable.

If the Launcher container suddenly appeared onscreen without any transition, it would seem deceptively static. It would be hard for people to know that the

content is scrollable, and that there are more options available than those currently displayed. It is very valuable to use transitions as a way to visually hint at functionality.

c. Guidance animations

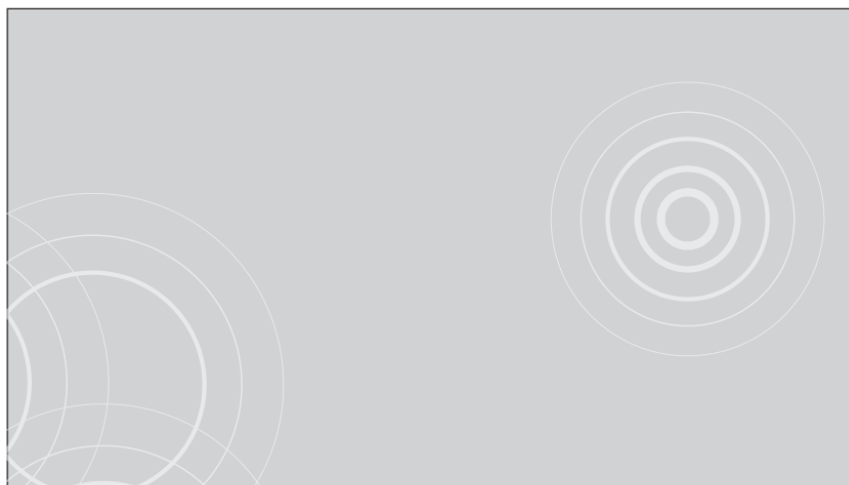
Guidance animations teach people how to perform specific touch interactions to accomplish a task. They communicate the way a particular button icon or piece of content should be used and give people feedback to visually explain a task.



For example, the Access Points use guidance animations to teach people a specific direct touch interaction. When someone touches the static Access Points, a visual indicator and action word animate out of the Access Point. If the person does nothing, the visual indication fades out. If the person slides the Access Point or taps the action word, they navigate to a different part of the Surface experience.

d. Ambient animations

Ambient animations are subtle animations used to indicate that the system is alive, awake, and ready for use. They can help reinforce brand and application personality. Ambient animations are purposefully unassuming and refined using subtle and natural effects.



Subtle ambient movement in the background can give a Surface experience a lively quality. Use ambient animations in your application to ensure people that the application is responsive and working.

Section 3.3:

SOUND DESIGN GUIDELINES

Sound design for Surface should draw heavily from the Surface design principles. Adhere to the following guidelines when developing or using sounds for Surface applications.

1. Make sounds simple and subtle
2. Ensure sounds are authentically digital
3. Use sound judiciously. Sound is often difficult to hear in public locations. If sounds are overbearing or annoying people will mute the speakers or remove the application.
4. Make sure all application sounds are at a level (volume) consistent with the sounds in the Surface Shell and Launcher. If sounds suddenly spike in the middle of an experience people will mute the speakers or remove the application.

LANGUAGE & TEXT DESIGN GUIDELINES

Surface experiences typically require little interface text, but this will vary with the type of application you are developing. Text should be content oriented and used judiciously.

Using the Surface design principles to influence the personality, terminology, capitalization, and punctuation of text will add an extra layer of polish to applications and significantly enhance usability.

1. Personality and disposition

The personality of text in Surface applications should be lively, clear, and consistent. It should not include soulless, jargon filled, incomprehensible computer-like language that prevents people from enjoying themselves – or asks them nonsensical questions in cryptic dialogue boxes. The disposition of text in Surface applications should be friendly, lighthearted, and empathetic. Application text should not be rigid, snobby, or uninformative.

An easy way to see if the text in your application has the right personality and disposition is to see if it sounds like a friend assisting another friend with a task or objective. For example a message like “Error Code: C82FF4B696” or “Critical data not found” is likely to confuse, frustrate and even anger people. A message with the right personality and disposition might look something more like, “There is some info missing here. Please put your name in the text box to continue.”

2. Capitalization

Surface displays text in lowercase and all caps layouts in many places, but also uses title caps, where the first and last words of the phrase and all words in between are capitalized, and sentence caps where only the first word of a sentence is capitalized.

Title caps exceptions are articles (a, an, the), coordinating conjunctions (and, but, for, not, or, so, yet), and prepositions with four or fewer letters (at, for, in, into). For example, “Neon Tetras in My Fish Tank.”

Sentence caps exceptions are words that are normally capitalized in text such as proper nouns or feature names. For example, “I want to visit Mt. Rainier in the springtime.”

Maintain consistent capitalization practices to prevent disjointed or jagged reading experiences.

- a. Use lowercase for:
 - i. Content titles
 - ii. List titles
 - iii. List items
 - iv. Interface control labels
 - v. Example text that appears in search and text boxes
- b. Use sentence caps for:
 - i. Check box and radio buttons labels
 - ii. Progress indicators
 - iii. Status, notification, and explanatory text
 - iv. Toggle switches
- c. Use all caps for:
 - i. Application titles
 - ii. Dates and times
 - iii. AM or PM

3. Punctuation

Use consistent punctuation to avoid confusion, clarify ambiguous text, and provide direct emphasis as needed.

- a. Ampersand (&)
 - i. Okay to use in settings or menu lists, for example: Date & Time; Clocks & Alarms.
- b. Colon (:)
 - i. Do not use a colon at the end of labels for controls such as text boxes, drop-down menus, and progress bars.
 - ii. Do not use a colon when the text box or drop-down list is embedded in a sentence or when the drop-down list appears in the main window.

- iii. Do not use a colon at the end of group headings or column headings.
- iv. Use a colon to introduce numbers or other variables, for example:
Percent Downloaded: XX%
- c. Ellipsis (...)
 - i. Use an ellipsis in progress indicator labels to indicate a continuing action, for example, when a person downloads a file. Even if there is a visual progress indicator, you still want to use the ellipsis.
 - ii. Do not use an ellipsis in headings.
 - iii. Do not use an ellipsis in button labels.
- d. End punctuation (. ? !)
 - i. Use end punctuation only in instructional text. Do not use end punctuation if instructional text appears in a label or button.
 - ii. Do not use a period at the end of option or check box text labels, even if the label is a sentence.
 - iii. Separate sentences with one space after the ending punctuation, not with two spaces.
 - iv. End a question with a question mark. Avoid phrasing labels as questions.
- e. Parenthesis ()
 - i. Avoid using parenthesis in applications if possible, but use a parenthesis if you need to include an acronym or other short piece of information.

Section 4.0:

SURFACE HARDWARE

This section discusses design considerations and guidelines that relate specifically to the Surface hardware.

Section 4.1:

INPUT METHODS

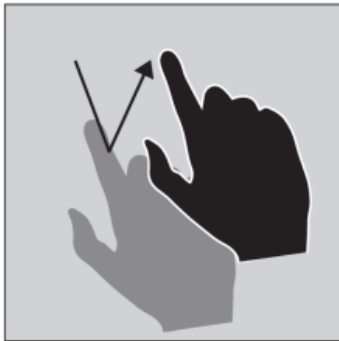
1. Vision based touch

The primary mode of interaction with Surface is touch. Surface is designed for touch from the ground up, and Surface applications should be written with direct touch interaction as the key driving force.

Fingers & blobs – Surface recognizes and differentiates between fingers and blobs. Fingers are identified when someone places their finger on the screen. The number of fingers, their position, and orientation – which direction they're pointed – are detected by the vision input system. Other items placed on the screen not identified as fingers or tags are classified as blobs. Basic size information is provided and an arbitrary orientation is assigned. Orientation values for blobs are generally not as reliable as fingers or tags.

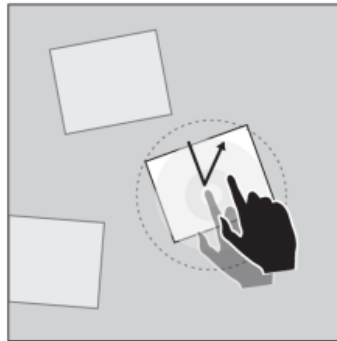
Touch interactions – The Surface SDK manipulation processor recognizes three discrete manipulations: move, rotate, & resize. The fact that there are only three manipulation gestures in the Surface SDK is a technical fact, but from an interaction perspective, there are many different direct touch interactions that a person can use with these manipulations. The following illustrations highlight the variety of touch interactions that can be performed on a virtual object with one finger or several fingers, using one or more hands.

Tap – Press and then release

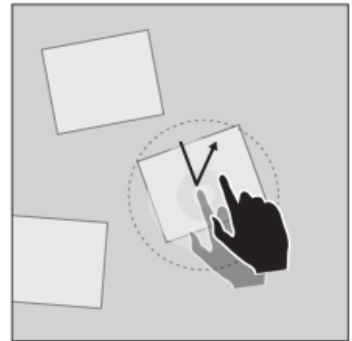


Tap

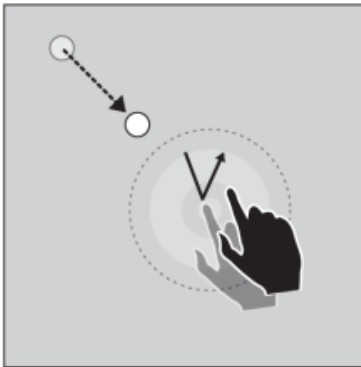
=



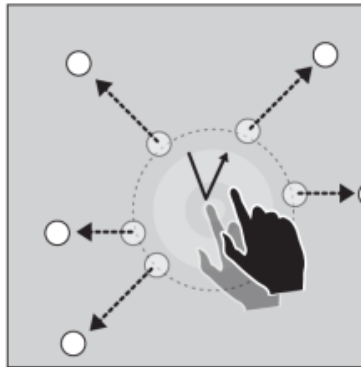
Tap to Select



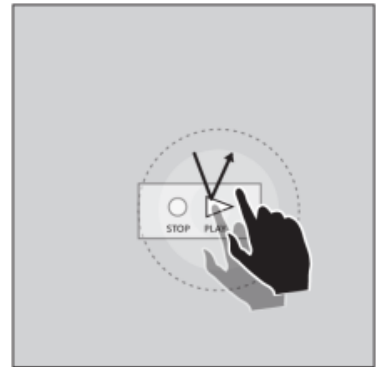
Tap to Deselect



Tap to Draw Near

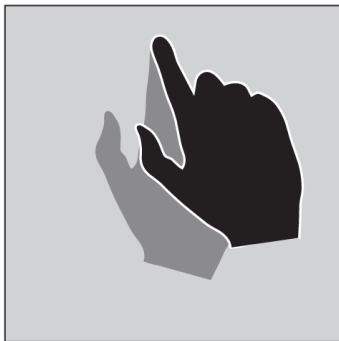


Tap to Scatter



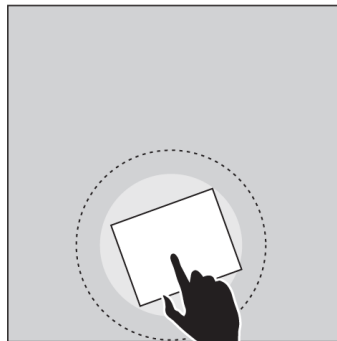
Tap to Activate Control

Hold – Press and then hold

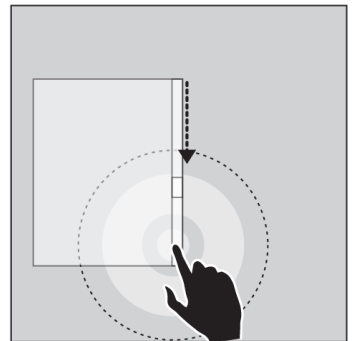


Hold

=

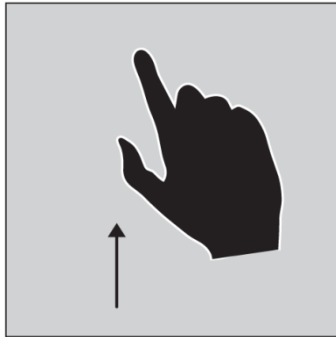


Hold to Select



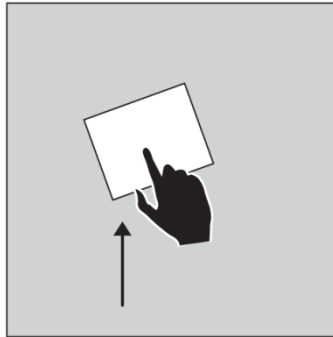
Hold to Autoscroll

Slide or push – Move the object under your finger with a sliding or pushing action

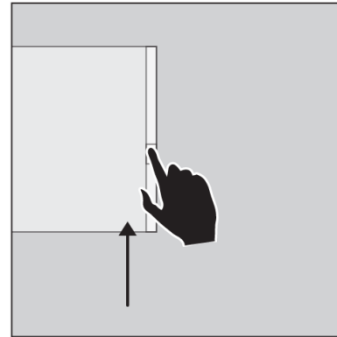


Slide or Push

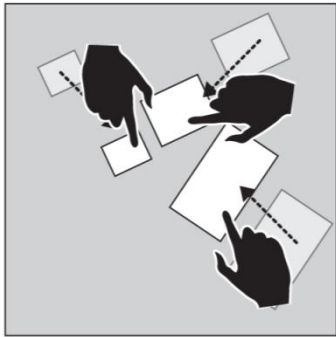
=



Slide or Push to Move

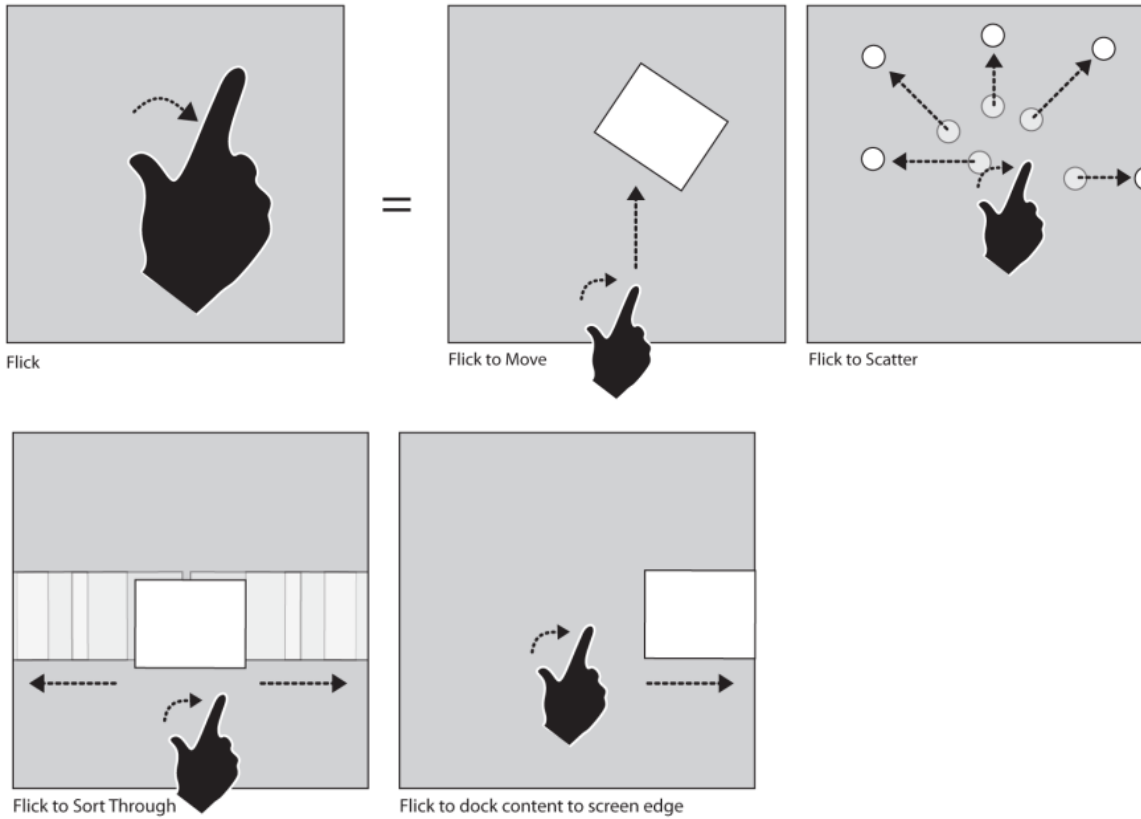


Slide or Push to Use Controls

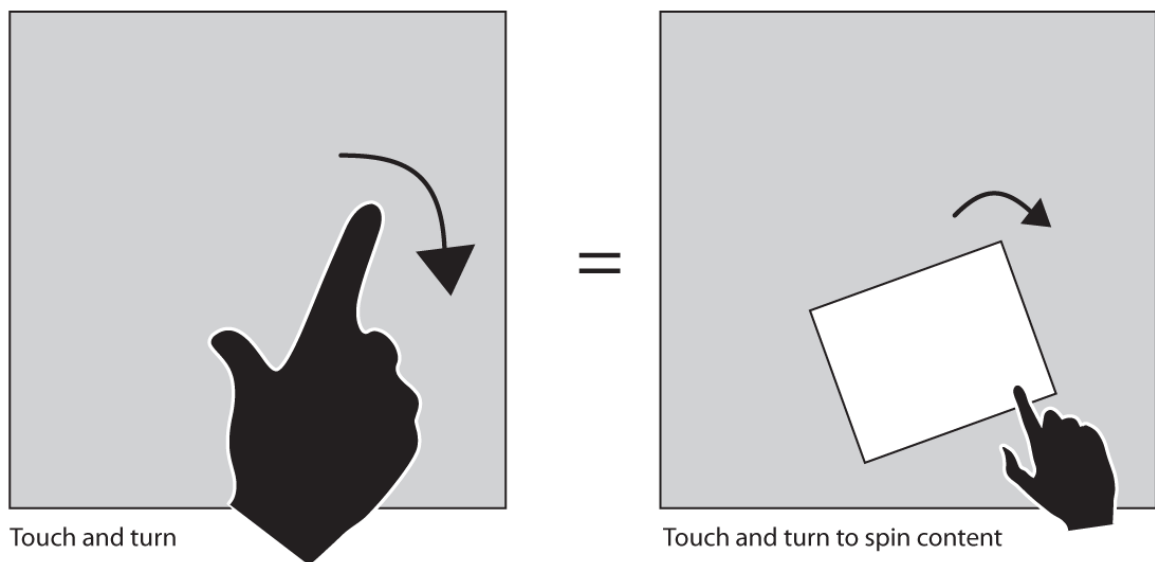


Slide or push to collect content

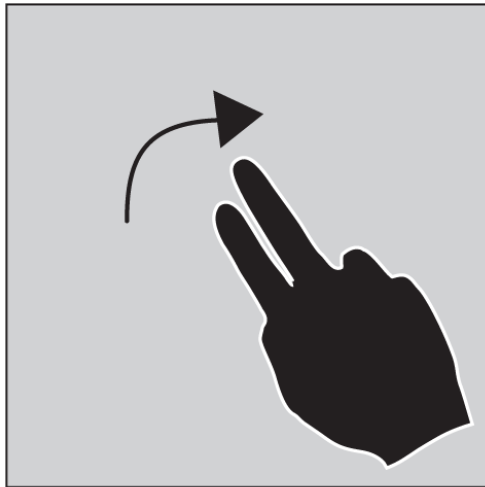
Flick – Press, slide quickly, and then release



Touch and turn – Slide your finger near the outside edge of a piece of content and rotate it around its center

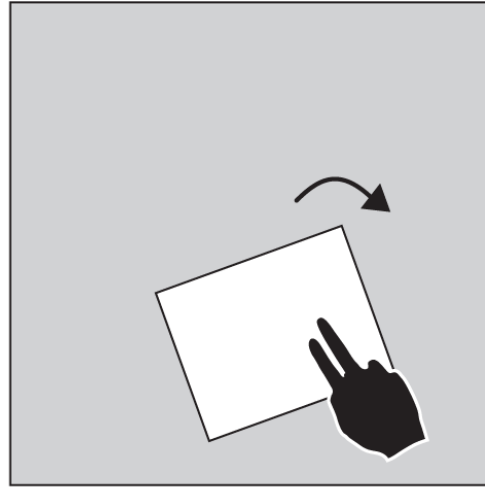


Spin – Twist quickly with two fingers to rotate the object



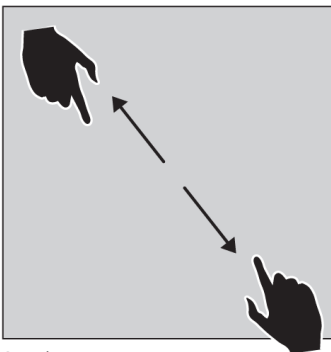
Spin

=



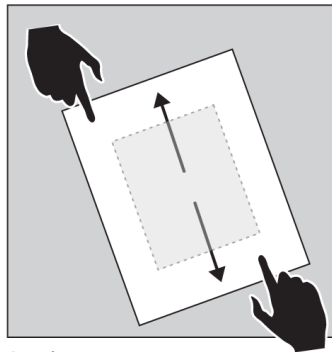
Spin to turn content

Stretch – Move fingers apart with two hands

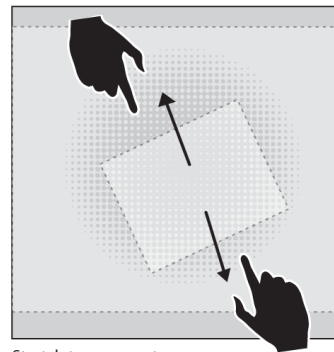


Stretch

=

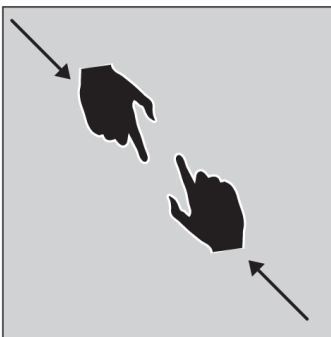


Stretch to resize



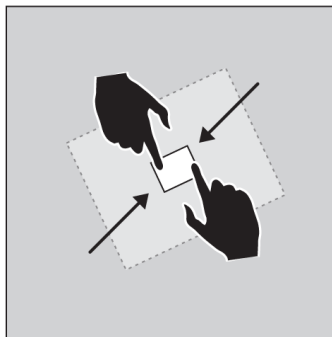
Stretch to zoom out

Shrink – Bring fingers together with two hands

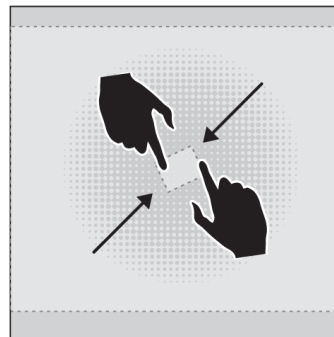


Shrink

=

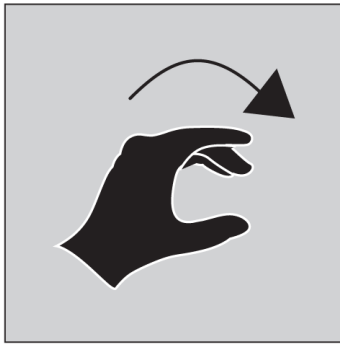


Shrink to reduce size



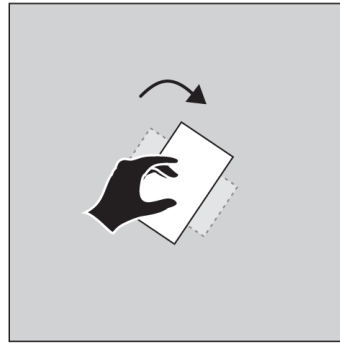
Shrink to zoom out

Twist – Twist the object with two or more fingers, like turning a knob or sheet of paper

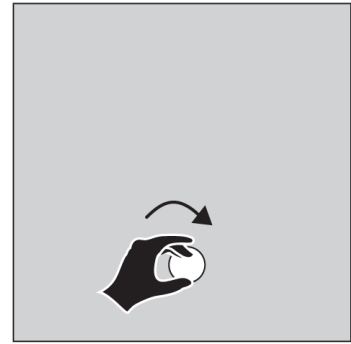


Twist

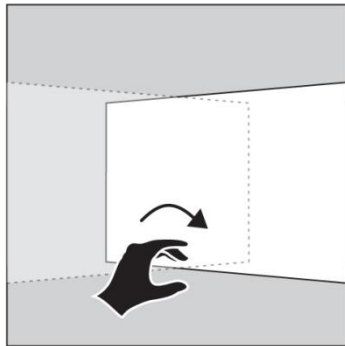
=



Twist to spin objects

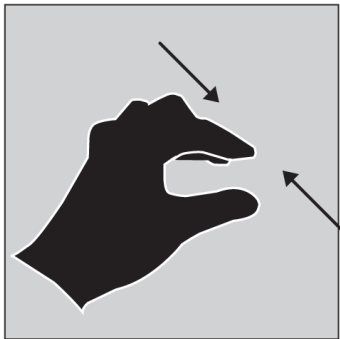


Twist to adjust control



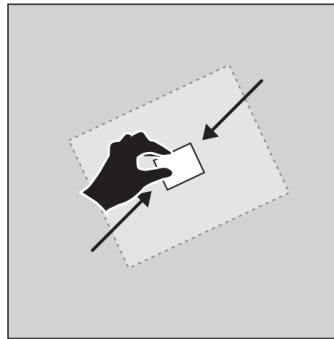
Twist to change perspective

Pinch – Move two fingers together on one hand

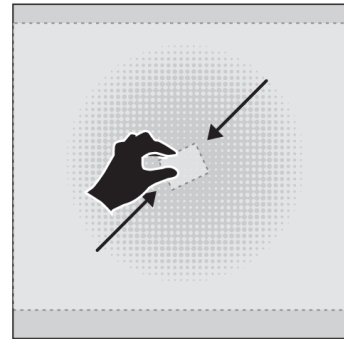


Pinch

=



Pinch to reduce size



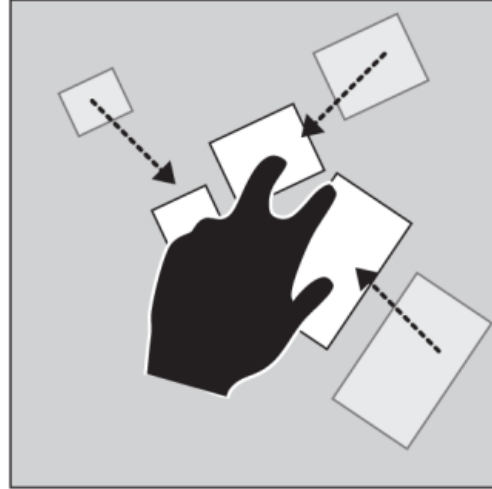
Pinch to zoom out

Squeeze – Move three or more fingers together on one hand

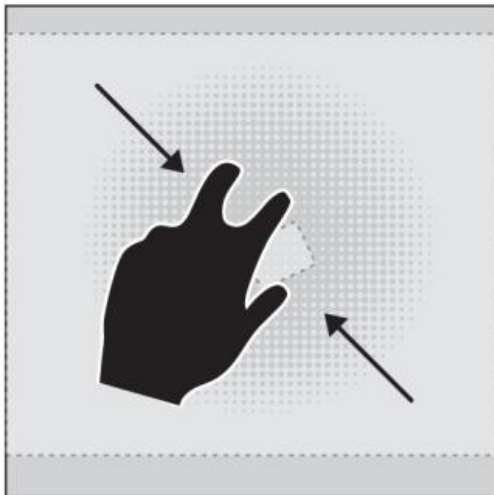


Squeeze

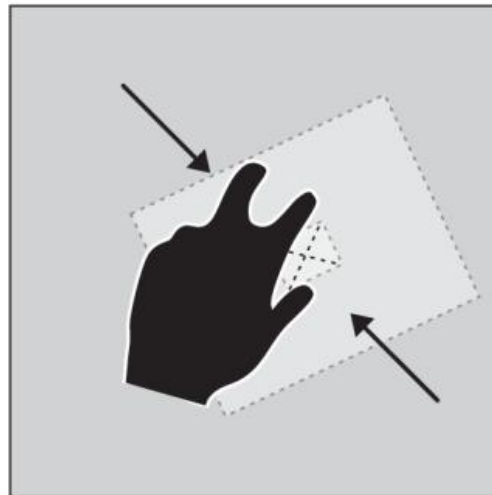
=



Squeeze to collect content

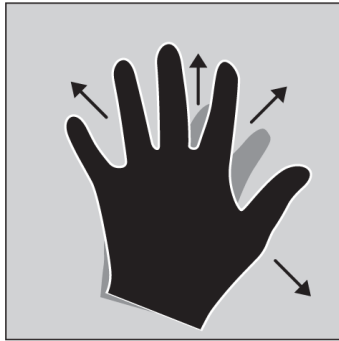


Squeeze to zoom out



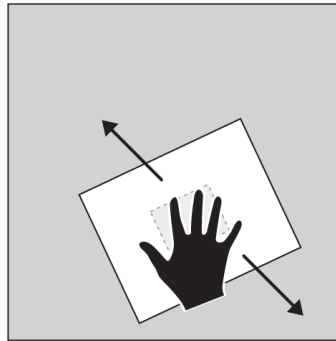
Squeeze to close content

Spread – Move fingers apart on one hand

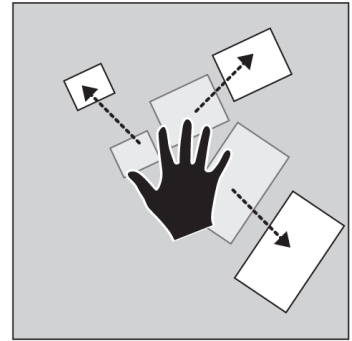


Spread

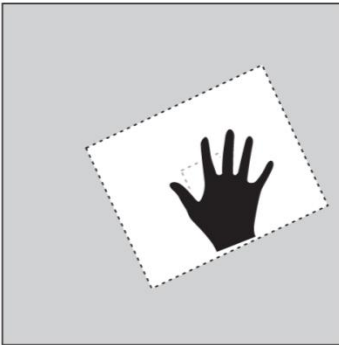
=



Spread to enlarge content

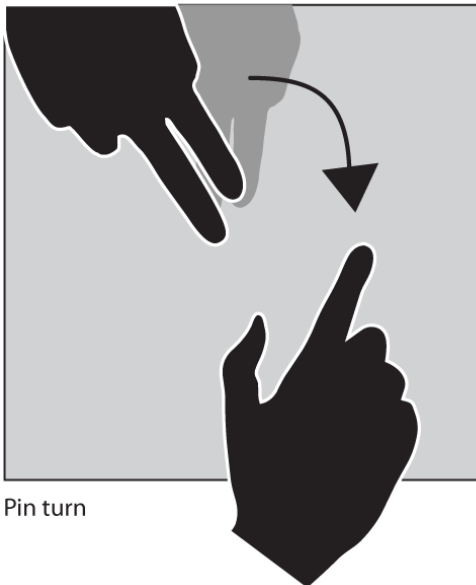


Spread to scatter content



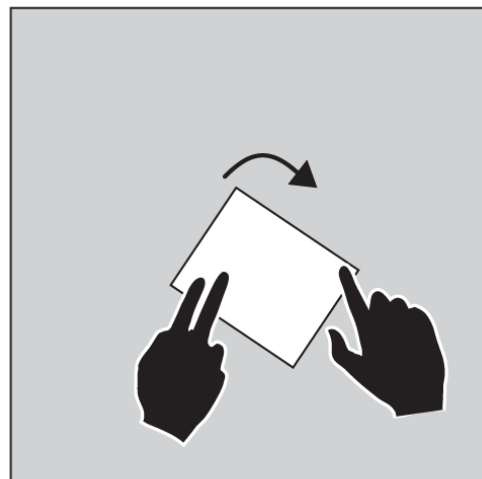
Spread to open content

Pin turn – Pin the object down with one finger while the other hand drags the object around the pinned point



Pin turn

=



Pin turn to rotate content

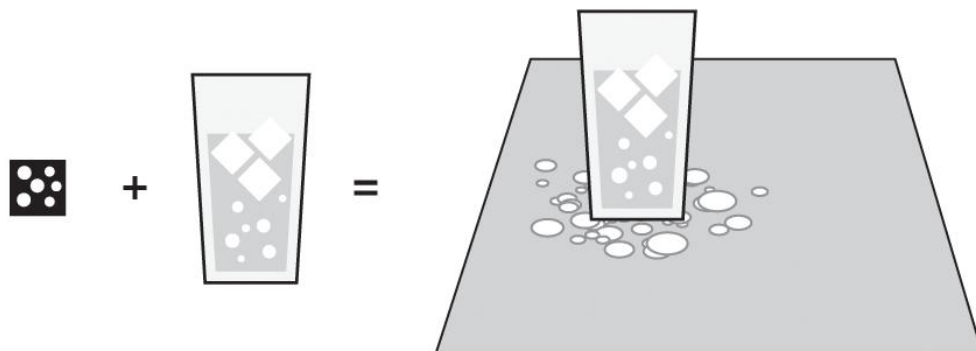
2. Objects

Surface not only supports touch interaction with fingers, it also supports a range of interactions with objects touching the screen. Using objects in Surface applications provides these unique benefits:

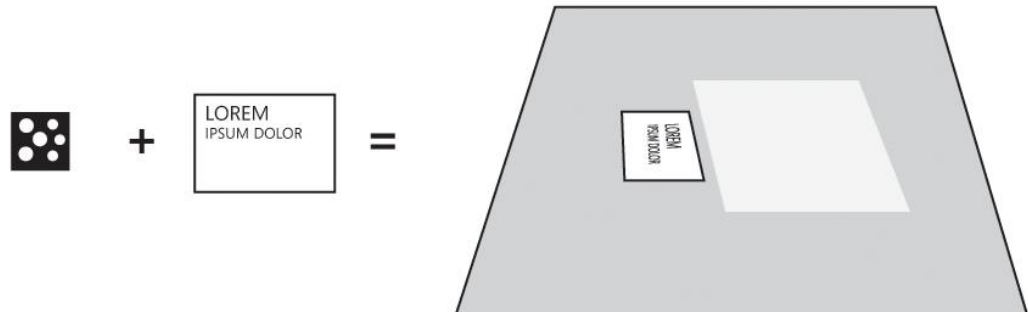
- Helps people easily share control of an application
- Helps the application identify who is using it and where they are physically located in relation to the screen
- Provides people intuitive ways to instantly understand and interact
- Provides a way to instantly link physical objects with digital content
- Provides a virtually endless number of discrete identifiers and identities
- Provides a way for people to take digital information with them
- Provides a seamless link between Surface and other mobile devices
- Creates magical, engaging experiences in ways not previously possible

The types of objects supported are explained in detail below.

Tags & orientation – A tagged object is a physical object that has a special tag with a pattern of dots affixed to it that Surface sees in infrared (IR) light wavelengths. IR light is not visible to human eyes, but Microsoft Surface hardware sees and recognizes the tag to perform the actions associated with the tag. For example, an application in a restaurant or bar could use a tag on the bottom of a glass so that when a customer puts the glass on the screen, bubbles appear around the bottom of the glass.



In another example, a tag can be used to display a menu or other information when a person places an object on the screen.



Tagged objects can also be dynamically associated with content in a particular application. This means that objects can be reused across applications without having to hardcode specific values. One application may prompt a restaurant server to place a tagged glass of orange juice down on a particular spot of the display. That application does not initially require a specific unique tag value, but the act placing that object down in a known spot lets the server identify to the application that the object placed there is a glass of orange juice. This can be repeated as needed; for example, a server might also add a tagged cup of coffee to the table in a specified spot. Once that tag is associated with the coffee, it can remain so until the customer dining experience is ended. If someone picks up the cups and places them back down on a different part of the screen, the Surface will still be able to identify them.

Object blob data, raw image & contact geometry – In addition to seeing special tags, Surface hardware can most objects placed on the screen, as long as they reflect back enough infrared light. Application developers can use the dimensions of objects which reflect basic shapes to their advantage; such as simple ellipses and rectangles. For example a coin placed on the screen with a fixed and known diameter can be easily recognized within the context of a particular application. A cube or cylinder of fixed dimensions might also be placed on the screen for a specific interaction without the need for a tag.

More advanced applications can use the raw image data provided by the vision input system to analyze contact geometry, the outline or silhouette of an object, to provide meaningful interaction with complex shapes. For example a developer can

use the outline of a human hand to create new types of direct touch interactions. Objects with unique shapes like stars, triangles, octagons and so forth could also provide special interactions without the need for tags. This type of object recognition does come with potential application development and performance costs, but is capable of providing experiences well beyond those of typical Surface applications.

Some Surface applications use the raw bitmap image from the sensors to identify objects and their purpose within the context of a particular application. This method generally differs because it goes beyond looking at basic shapes to identify an object. Using raw image may be a good method to capture a snapshot of an object and compare it against a database of previously captured images. This method can provide positive identification for much more complex objects.

Other applications may not even be concerned with identifying a complex object at all, but may instead use the vision input processors to capture an image, much like a photocopier or scanner, so that people using the Surface may view, edit, and interact with the image. This type of image capture can be limited if objects don't reflect a sufficient amount of infrared light. Application developers and designers should experiment with the properties of their objects and test them in real deployment locations early in the planning phase of development.

Transparent objects – Many objects on Surface literally light up when placed on the screen. Some of these objects contain clear tags on them which contain the same byte patterns as standard tags, but can't be seen by the human eye. This enables clear glass and plastic objects to be placed on the screen and have software driven images projected right through them.

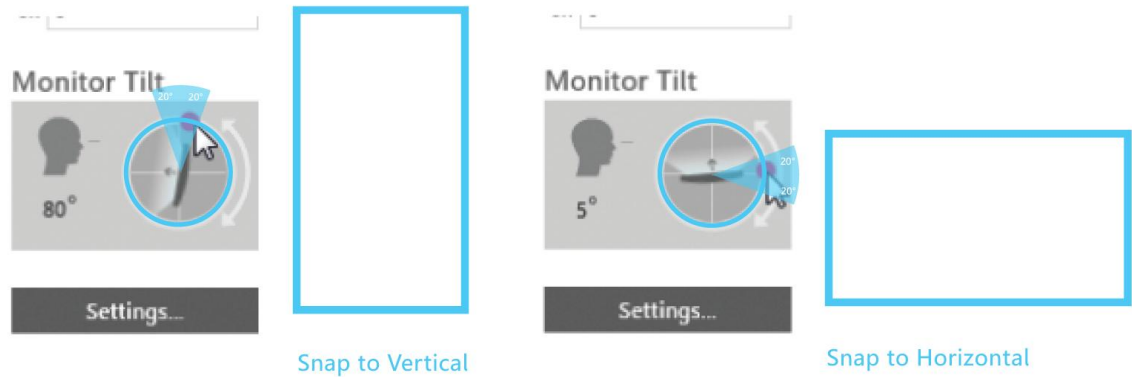
This can transform an inert plastic cube into a lively token, or a dinner glass into an interactive control object. Specialized objects can even use unique properties to alert the Surface when the liquid level in a glass has reached a certain level, or project animated visuals through objects using fiber optic elements.

These types of objects are often inexpensive ways to create unique experiences that other devices and platforms cannot deliver. The possibilities for using objects to interact with Surface applications are virtually limitless.

Section 4.2:

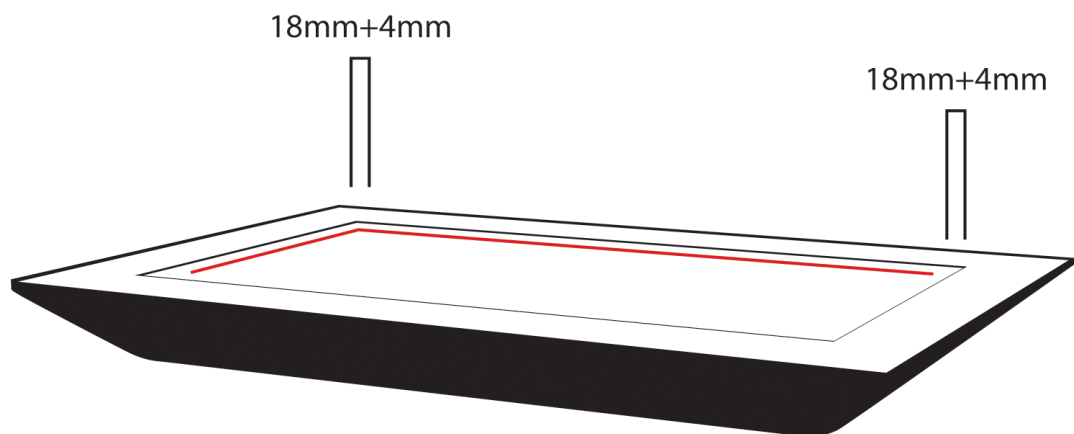
HARDWARE SPECIFIC CONSIDERATIONS

1. Tilt



Microsoft Surface hardware is designed for deployment in horizontal and vertical deployments. Sensors in the device will indicate the angle of tilt that the device is currently at. A horizontal device will report back a value close to 0° degrees and a vertical device will report back a value close to 90° degrees. Some devices may be mounted at an incline between 0° degrees and 90° degrees so it's important for application developers to keep this in mind when designing and testing their applications.

2. Bezel buffer



Application content should be designed with Surface hardware parameters in mind. The minimum size for touchable content should be at least 18mm x 18mm. Touchable content displayed at the screen edge should be inset at least 4mm to

ensure that finger and other contacts will be recognized by the sensors. The Surface Access Points in their rest state provide a good visual indicator of the minimum touch target size and distance to display interactive content away from the screen edge. Content docked at the screen edge should stick out at least 22mm from the screen edge in order to ensure that it is usable.

3. Connectivity

Surface hardware is equipped with Bluetooth and WiFi capabilities. This enables Surface to easily connect to a wireless network and interact with other devices. Surface may also be wired directly through its Ethernet port, which provides advance remote administration capabilities.

Many Surface applications also provide connectivity with mobile devices by using cloud based services and mobile applications. When combined with the vision input and tracking capabilities of the Surface hardware the interaction with mobile devices can be quite seamless. Surface is also equipped with HDMI in and out capabilities, as well as headphone, microphone, SD card and USB ports.

