User Experience Guidelines

User Interaction and Design Guidelines for Creating Microsoft Surface Applications

June 2, 2009

Contents

1 Introduction — 4

1.1 Touch and Multitouch — 4

1.2 Gesture, Manipulation, and Movements — 6

1.3 Objects — 9

2 Interaction Design Guidelines — 10

2.1 Interaction Principles — 10

2.2 Interaction Guidelines — 14

3 Visual Design Guidelines — 36

3.1 Visual Design Principles — 36

3.2 Visual Design Guidelines — 39

4 Interface Text Guidelines — 61

4.1 Language and Text Principles — 61

4.2 Language and Text Guidelines — 62

4.3 Text-Specific Component Guidelines — 70

1 Introduction

Microsoft Surface™ proposes a new relationship between humans and technology, one that unfolds intuitively through the natural human input of touch. This relationship is given space and form through the practice of, and attention to, design.

To design compelling Microsoft Surface experiences, you must think differently about the user interface (UI) than you do when creating traditional graphical user interface (GUI) products or Web experiences. In particular, if you are accustomed to working in the mouse-based world of GUI applications, there are many issues to consider when building applications for multitouch systems, particularly Microsoft Surface. For example, because Microsoft Surface is a multiuser system, you cannot simply enable users to click a tool that is applied to all contacts on a Microsoft Surface unit because one user's actions can interfere with the actions of another user.

This document aims to help the developer and designer create applications and user experiences that best exploit multitouch interaction principles through quality, multitouch-oriented design. The second chapter of this document describes the core principles for optimal user-interaction paradigms in multitouch applications developed for Microsoft Surface, and provides basic guidelines for when and how to employ those principles. Subsequent chapters provide the principles for visual and textual design along with practical guidelines for implementing them.

This document represents what the Microsoft Surface team has learned over the last several years. The design guidelines presented here are not exact specifications of Microsoft Surface applications. Instead, the guidelines include approaches to design great Microsoft Surface experiences so you can create applications that take full advantage of the Microsoft Surface promise of a natural user experience.

1.1 Touch and Multitouch

The primary mode of user interaction with Microsoft Surface is "natural" and "intuitive" touch. A good Microsoft Surface application offers a fun, engaging, and visceral experience without any chance of intimidation. Regardless of whether the application is a game, one that involves a task (such as ordering food from a menu or making a reservation or ordering tickets), or just a playful or entertaining experience (such as the water attract application or manipulating photos), the user's finger goes right to the Microsoft Surface screen and the user simply "knows" what to do.

Interacting with Microsoft Surface is as simple as touching. Users gain confidence and gradually try more interactions, from one finger to multiple fingers. Microsoft Surface is a multitouch platform; moreover, it can be approached and used from any side. While more traditional applications with a specific orientation to one side of the unit are possible and indeed valuable, the real uniqueness of Microsoft Surface is that can offer a 360-degree user interface without any real top, bottom, left, or right.

Multitouch also means that two, three, or more people can use it at the same time. Further, Microsoft Surface can respond to physical objects placed on it. Microsoft Surface offers a tremendous opportunity for creating social interaction among users.

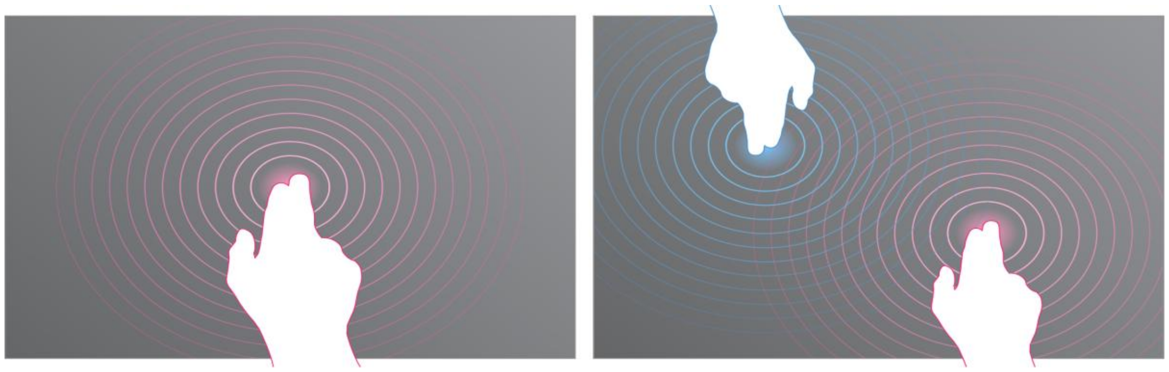


Figure 1-1: Microsoft Surface can respond to one or multiple touches

Use an entire hand to interact with Microsoft Surface, as it can sense and respond to large contacts.

Figure 1-2: Microsoft Surface responds to a hand and other large object as a single contact

Use gestures to interact with Microsoft Surface. Brush or move a finger or hand along the screen and see that it responds to more than just a stationary touch. Microsoft Surface is social after a second or third person joins in.

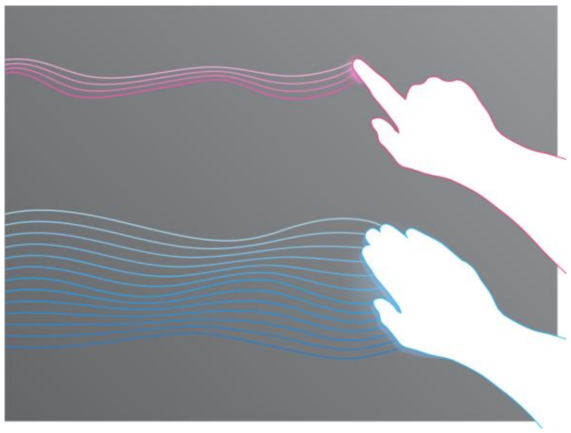


Figure 1-3: Microsoft Surface is social after a second or third person joins in

Microsoft Surface responds to objects. In the default water attract application, as soon as someone brings an object into contact with the screen, water ripples will bounce off the object.

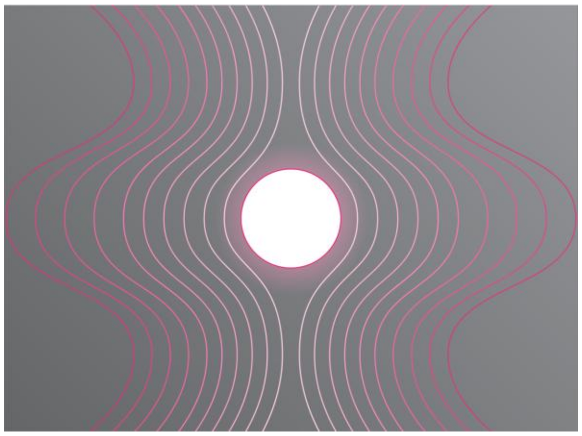


Figure 1-4: Microsoft Surface can respond to objects visually

Keep these key points in mind:

- Microsoft Surface can recognize and respond to more than 50 discrete simultaneous contacts.
- Depending on the goal, a Microsoft Surface application can be designed for one person or for multiple people to use simultaneously. (In many cases, the same application can do both.)
- Similarly, depending on the goal, a Microsoft Surface application can have a top and bottom but can also have a 360-degree orientation in which users can approach the unit and use the application from any side.
- An application that most fully exploits the uniqueness and potential of Microsoft Surface is a multiuser experience with a 360-degree orientation.

1.2 Gesture, Manipulation, and Movements

Let's clarify the Microsoft Surface terminology. Readers of this document who will design Microsoft Surface applications must understand the different things the system can do and how we will refer to them.

1.2.1 Three Manipulation Gestures

Microsoft Surface recognizes and responds to one kind of touch; this is referred to variously as a *manipulation,* a *gesture*, or a *manipulation gesture*.

A manipulation gesture is when you put a finger (or object) down on a virtual object and cause a change to that object while your finger or the object retains its position in relation to the object.

For example, assume you have a photo of three people standing next to each other. You touch the head of the person in the middle and slide your finger to right. The photo moves to the right, and the middle person's head stays under your finger. At the same time, if you touch one finger to the upper-right corner of the photo and another finger to the bottom-left corner, you can then rotate the photo by sliding your fingers in either a clockwise or counterclockwise direction. The photo moves and your fingers stay in the same places on the photo.

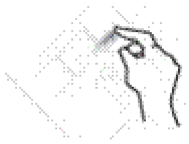
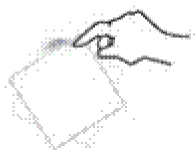
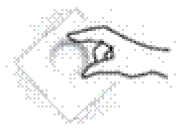
In traditional applications, a *system gesture* is one in which the system recognizes a contact and runs a command based on that contact. A system gesture causes changes to the system state without a change in the visual representation under a user's finger or object. For example, a user draws a question mark to open Help. Or, a user puts a physical object on the screen and causes virtual objects to scatter—this is a change in system state even though it has an associated visual change. *There are no examples of system gestures in the Microsoft Surface platform today*.

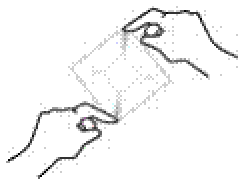
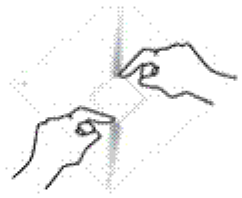
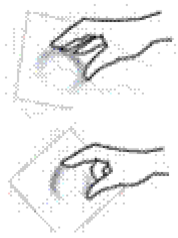
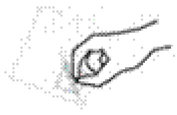
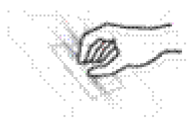
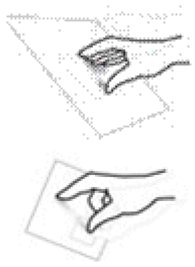
Microsoft Surface recognizes three discrete manipulation gestures:

- Move
- Rotate
- Resize

1.2.2 Movements

The fact that there are only three manipulation gestures in the Microsoft Surface SDK is a technical fact. From a design and user-interaction perspective, however, there are many different movements that a user can make with those three manipulation gestures. This opens up a world of interactive design possibilities. The following table shows a variety of movements that can be done on a virtual object with one finger and multiple fingers (sometimes using either one or both hands).

Movement	**Description**	
	Tap	Press and then release
	Slide or push	Move the object under your finger with a sliding or pushing action
	Flick	Press, slide quickly, and then release
	Touch-and-turn	Slide your finger on the content around a point of the content; you can also fake this carefully with manipulations in ScatterView
	Spin	Twist quickly to rotate the object

	Pull apart Stretch	Pull fingers apart on two hands
	Push together Shrink	Bring fingers together on two hands
	Twist	Twist the object with two or more fingers, like turning a knob or paper
	Pinch	Bring two fingers together on one hand
	Squeeze	Bring three or more fingers together on one hand
	Spread	Pull fingers apart on one hand
	Pin turn	Pin the object in place with one finger while the other finger drags the object around the pinned point

1.3 Objects

Microsoft Surface can respond to physical objects. There are two types of objects: untagged and tagged.

- An *untagged object* is literally any physical object without the special Microsoft Surface tag. It produces in the system what is called a “blob”: the system knows a “thing” is touching it, but does not know its precise shape. It is not possible to use the SDK to recognize objects that are not tagged.
- A *tagged object* is a physical object that has a special tag with a pattern of dots visible in infrared affixed to it. Microsoft Surface reads the tag and performs whatever actions are programmed for that tag. For an example, an application in a restaurant or bar could use a tag on the bottom of a glass so that when the glass is placed on the unit, condensation bubbles appear around the bottom of the glass. In another example, a tag can be used to display a menu or description when an object is placed on the unit.

2 Interaction Design Guidelines

Interaction design defines the interplay of the software experience with users' behaviors, responses, and gestures. In other words, interaction design principles are geared toward maximizing the user experience by facilitating the ease with which they can learn, become immersed in, and enjoy the software experience. While closely aligned with purely visual graphic design, audio design, and textual design, interaction design provides an overall enveloping focus on how well and naturally users interact with technology.

Microsoft Surface is a major new development in the area of Natural User Interface (NUI). More closely aligned to sophisticated video games that create an entire universe in which users become totally immersed than to traditional keyboard/mouse productivity software whose only goal is to complete and accomplish tasks by producing and/or analyzing data, Microsoft Surface adds the crucial dimension of direct manipulation with fingers and objects on a large screen that can accommodate multiple users simultaneously. While Microsoft Surface applications can have a discrete top and bottom orientation, the potential of Microsoft Surface is most fully realized by applications with a 360-degree orientation that can be used by a number of people at the same time.

The first section of this chapter describes the theory behind the interaction design principles. The second section provides actionable guidelines for realizing those principles in Microsoft Surface applications.

2.1 Interaction Principles

This section gives a high-level description of the eight user-interaction design principles behind the development of successful Microsoft Surface applications. There is a great deal of interconnectedness among the principles; how much emphasis you place on any given set of animating principles depends significantly on the application you're creating. For example, an application that a hotel might use to help people locate places of interest and plan a walking tour should be responsive and visually engaging but is primarily a tool for gathering information. A more playful application, on the other hand, such as the Microsoft Surface Photos application, will place more emphasis on engaging interactions and creating delight. It is therefore important to understand the nuances of all the principles and how they work with and complement each other.

2.1.1 Seamless

Seamless experiences require users to be mentally and emotionally immersed so they fearlessly commit to new experiences. You create seamless experiences by suspending the users' sense of disbelief. The *suspension of disbelief* refers to a person's willingness to accept something as true or sufficiently real even if it is fantastic or impossible in the real world. The application creates a self-contained, immersive world that simulates a living, breathing environment.

Microsoft Surface experiences suspend disbelief by mimicking real-world objects and using virtual-world capabilities to extend the objects beyond what is possible in the real world. Imagine a Microsoft Surface application that initially appears as a globe that you can spin by flicking it with your finger. You touch a location on the globe to zoom in closer. Each touch zooms in further until you see points of interest that you touch to create a personalized itinerary.

To suspend disbelief successfully, erase the line between the physical and virtual worlds in a way that is seamless and in which the performance of the technology is flawless. A Microsoft Surface experience must respond continuously to fingers and physical objects that are placed on it and must immerse users in a better-than-life experience. It must respond continuously to fingers and physical objects by displaying information on the screen. For example, if a user places a cocktail glass on a Microsoft Surface screen in a lounge, the user interface should respond to it by surrounding the glass and displaying the name and ingredients of the cocktail (assuming that the glass has a drink-specific byte tag or identity tag).

2.1.2 Social

Standard GUI applications explicitly create social barriers because of their input and output methods. For example, experiences are inherently single-person when users have only one mouse, one keyboard, and no touch screen. In contrast, Microsoft Surface accepts multiperson input, so any number of people can gather around one Microsoft Surface unit and play the same instance of a game or manipulate different photos at the same time. Microsoft Surface can elevate activities from a solitary experience to a social experience.

The social experience is not limited to the interactions between people and the Microsoft Surface interface. Communication (that is, interaction) between an individual user and the Microsoft Surface unit is sufficiently natural and intuitive as to be totally unobtrusive; and the more unobtrusive the communication between an individual and the Microsoft Surface unit, the more communication that happens between the people around the unit. People focus more on each other than on the computer, so the computer becomes secondary to the people that are using it.

You can reuse cooperative techniques from video game design in Microsoft Surface experiences to make them more engaging, fun, and social. When conceiving and designing Microsoft Surface applications for multiple users, consider how users manipulate objects and complete tasks simultaneously; how the Microsoft Surface experience is oriented to people who are themselves are arranged around the four sides of the unit.

2.1.3 Spatial

Traditional GUI interaction models are flat, planar, and two-dimensional (2D). You can use some *two-and-a-half dimensional* (2.5-D) techniques, such as skewing planes, adding shadows, and overlapping elements, to give depth to some objects. However, Microsoft Surface interaction models go beyond a simple plane to provide depth, encourage immersion, and make objects appear to have volume or take on real-world, three-dimensional (3D) behaviors so people can navigate spatially in all dimensions.

Not all applications, however, benefit from 3D environments. Sometimes 3D environments are disorienting and overly complex, but your application's behaviors, transitions, and navigation should nonetheless always consider the z-axis. For example, photos and videos in the Photos application are inherently resting on a flat canvas, but they rise to the surface when users touch them to give the feeling of depth and realism.

Microsoft Surface experiences represent objects volumetrically and leverage a user's depth perception and spatial memory. Environments can extend well off-screen, and users can drag the environment around to relocate content. Objects can be stacked in 3D space, using depth to sort, distribute, or focus on content. As long as users can use gestures to navigate the environment and orient themselves, they can create a mental model of the space, its content, and the gestures that they need to access that content without needing to see it all on-screen. Users naturally develop associations between what they want to do (for example, play a game) and where they do it (for example, in a game application) from memory-triggered context.

2.1.4 Super-realism

Because touch is inherently physical, it creates a sense of direct interaction with, and control of, technology. You can create more fluid, natural experiences by mimicking real-world physical interactions and augmenting them beyond what is possible in the real world. *Super-realism* pushes beyond what is physically natural so that experiences do more than what is possible in the real world. In this way, super-realism is natural and direct, but not purely literal.

For example, a Microsoft Surface experience enables a user to touch a photo and push it around, just like in the real world.

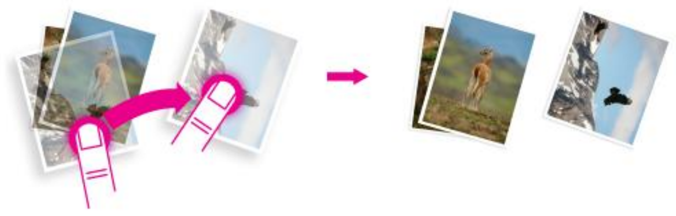


Figure 2-1. A user touches a photo and pushes it around

However, *unlike* the real world, the Microsoft Surface experience enables the user to scale the photo by using a two-finger gesture.

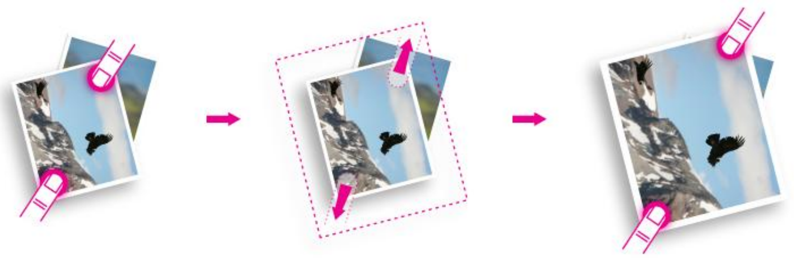


Figure 2-2. A user scales a photo using two fingers

The objects in a Microsoft Surface experience have a "better than real" quality that echoes the physical nature of real-world objects. Like the real world, direct manipulation causes an immediate response to all actions. Every time a finger, hand, or physical object interacts with a Microsoft Surface experience, a response occurs to make the experience feel "alive" and natural. To design such an experience, you should make objects do more than what is possible in the real world and not just replicate natural objects and interactions so that the objects stand apart from the real world. Super realistic objects with super realistic capabilities give people a sense of control that is not possible with other methods of interaction.

To create natural interactions, create the base of the interactions in the real world and then extend them in intuitive ways. To create super-realistic interactions, leverage the possibilities of virtual objects in digital environments to exceed what is possible in the real world.

2.1.5 Contextual Environments

Microsoft Surface experiences respond to the people and objects that touch it and react based on the context of the internal and the external environments, changing to what people are doing based on where they are in the experience and their behaviors. (In the video game industry, this reaction is known as *causality*.)

Microsoft Surface experiences should respond to people's actions in a way that clearly results from their context and behaviors so they can understand that what happens is a direct result of their actions, and so they can learn to anticipate and control the consequences of any action. Without this type of reaction, the experience feels random and users do not know how to make changes in the environment.

2.1.6 Scaffolding

The Microsoft Surface experience should offer fewer choices instead of superfluous features, and should create one simple but compelling solution that is richer, more fun, and more rewarding. For example, there might be only one way to print, but this path might involve literally throwing photos onto a visual representation of a printer.

To help users discover a solution through a rich and pleasurable journey, use *scaffolding* in your Microsoft Surface experiences. Scaffolding is a teaching method that breaks down bigger challenges (such as "How does this whole system work?") and focuses on smaller problem-solving challenges (such as "How do I initiate this one action?") through specific prompts, hints, and leading questions. Scaffolding provides supportive structures and moments that encourage active exploration instead of memorization and repetitions. You can use scaffolding to structure Microsoft Surface experiences by optimizing tasks into small, self-evident steps that maximize the visual or emotional reward and by simplifying moment-to-moment decision-making.

As part of scaffolding, present users with only the fewest reasonable choices at a given moment because users feel confused and overwhelmed when there are too many choices to consider. With fewer choices, the experience simplifies decision-making, discloses information or required choices over time, and simplifies a user's thought and action, so the experience is easier to use and enjoy.

However, simplicity need not mean simplistic; simple processes and tasks can be incredibly rich and powerful. Microsoft Surface experiences should demonstrate robust results that do not require complex procedures.

2.1.7 Performance Aesthetics

Video games such as *Myst*® and its sequel *Riven*® motivate users to simply explore the beauty of the game's world. Microsoft Surface experiences should also have visual beauty, but must also include the aesthetic of performance. Due to the physical nature of touch, gesture, and direct manipulation, a Microsoft Surface application must include seamlessness between objects that touch the surface and information that appears on the screen. An application must not include performance lags or hiccups that suspend belief in the visual fidelity, frame rate, or acoustic fidelity. Microsoft Surface experiences should be flawless, responsive, and immediate to make the experience feel accurate, smooth, and natural.

The aesthetics of beauty *and* performance in Microsoft Surface experiences combine to enable users to engage in multisensory interactions that create emotionally engaging experiences. These emotionally engaging experiences enable users to imagine, discover, and experiment with the world and establish important social boundaries, relationships, and rules.

To create visual beauty and performance aesthetics in Microsoft Surface experiences, your application must appeal to users' senses:

- *Sight* and *sound*. Create highly crafted visual representations of content, motion, and sound.
- *Imagination*. Enable users to discover new places, things, or effects and then stimulate their imagination with visually compelling cues.
- *Challenge*. Include challenges, but not daunting problems, for users to solve as they find solutions to the visual cues to provide an adrenaline rush.
- *Pacing*. Enable users to complete well-paced decision-making. Each user should have a set of rules, a context, and a goal, and then each user can find the most effective way to reach the goal.
- *Immersion*. Create experiences that users can deeply engage in, so they can enjoy "escaping" to this environment and exploring its systems, rules, culture, and space.

2.1.8 Direct Manipulation

The Microsoft Surface experience responds to touch and direct manipulation, whether by one finger, many fingers, or physical objects. For example, users can explore a music library by flipping through album covers, or play a chess game by using physical game pieces. The actions become temporal and unmediated in the way that content is manipulated.

2.2 Interaction Guidelines

The following guidelines are designed to provide developers with practical ways to embody the user interaction design principles discussed above; to turn principle into action, so to speak.

Each set of guidelines is comprised of three categories:

- The ***must*** category provides the minimum requirements to adhere to the design principle. The *must* guidelines also describe characteristics that are required by the Microsoft Surface application certification process. Specific requirements are cited in each guideline when appropriate.
- The ***should*** category describes guidelines that provide excellent experiences for users and that you can implement at a relatively low cost by using Microsoft Surface tools.
- The ***could*** category lists guidelines that we recommend so that your application provides a more complete, desirable, and fulfilling user experience. However, these guidelines might also cost more so you should prioritize between them. These guidelines might also apply only to particular application scenarios.

2.2.1 Make Virtual Objects Behave Like Physical Objects

A fundamental way to achieve a complete, believable universe is by making objects on the screen behave the same way that objects in the real world behave. You can add subtle physics in performing motions, inertia of movement, and natural-feeling collisions to help create the sense of virtual reality.

	Make Virtual Objects Behave Like Physical Objects
Must	Make every transition fluid. Every object and visible property change must smoothly animate and transition into and out of existence, or between changes. Nothing should abruptly appear or disappear. [Application Certification #01]
Should	Create transition animations that communicate state and relationship changes, contribute to a consistent interaction paradigm, and give a personality to the application.
	Mimic the real world in your transitions by using notions such as mass, acceleration, friction, viscosity, and gravity. The **ScatterView** controls in the Microsoft Surface SDK enable you to create these effects.
	Make sure the controls for starting and ending and for major state changes are always visible. This visibility contrasts with systems that embed major functions within menus.
	Break from "real-world" behavior to match user intent. All interaction metaphors start with physical manipulation and then extend it. For more information, see the "Super-realism" and "Scaffolding" sections.
Could	Consider what advanced expert functionality you want to enable in addition to natural interactions. Provide a mechanism that extends natural behavior to transition the user from a novice to an expert. For more information, see the "Scaffolding" section.

2.2.2 Use Tagged Objects

You can affix either of two kinds of tags to a physical object to elicit a specific response. A *tag* is a special pattern of dots. The tag consists of a geometric arrangement of infrared reflective and absorbing areas. There are two types of tags: *byte tags* (which encode an 8-bit number) and *identity tags* (which encode a 64-bit number). For more information about tags, see the Microsoft Surface SDK Help documentation.

When a tagged object is placed on a Microsoft Surface screen, the vision system reads the tag and determines its value, location, and orientation. The vision system also interprets any other portion of the object that reflects infrared (IR) light as a contact and sends its information to the application. You must then create a visual response that is appropriate to that object.

	Use Tagged Objects
Must	There are no certification requirements that apply specifically to the reaction to tagged objects. The same requirements that apply to finger and blob contacts (see "Untagged Objects" below) in general also apply to contacts from tagged objects.
Should	Respond to *every* object and contact, regardless of whether or not it has a tag affixed. Even if an object is not interactive, the experience still provides subtle feedback that acknowledges the object so users don't wonder if the object is broken or malfunctioning.
	Respond *immediately* to the presence of tagged objects. This immediate response blurs the line between the real and the virtual.

	Consistently apply tags to the same physical location on objects, so the vision system can infer the location and shape of the object from the tag.
	Use non-IR reflective objects for tagging, so that the reflective portions of the object do not generate contacts within your application.
	If IR-reflective objects are required, use your knowledge of the tag location and shape of the object to filter (ignore) these contacts.
	Use byte tags to identify a type of object (for example, *a* mobile phone of model # X-72), and use identity tags to identify a particular object (for example, *the* mobile phone of model # X-72 with telephone number 206-555-0100).
	For identity tags, make the reaction consistent across object types, and particular to the object.
	Clearly connect the effects of objects with the objects.
Could	Respond to a particular object and seamlessly extend it into the virtual world. Connect the user interface with the object itself, as if the virtual portion has extended out of the object.
	Use physical objects for input that requires a higher degree of freedom because the vision system can more consistently detect the orientation of tags than that of fingers or blobs.

2.2.3 Use Untagged Objects

An untagged object is referred to as a *blob*. Microsoft Surface can detect IR-reflective objects that are placed on the screen. In many cases, these objects are identified as multiple contacts (identified as either blobs or fingers) for each contiguous IR-reflective portion of the object. The vision system cannot determine if adjacent blobs are part of the same physical object and cannot identify physical objects that do not have tags. For your application, contacts from untagged objects are the same as contacts from other objects that register as blobs, such as an entire hand that is placed down on the Microsoft Surface screen.

	Use Untagged Objects
Must	There are no certification requirements that apply specifically to the reaction to untagged objects. The same requirements that apply to finger and blob contacts in general also apply to contacts from untagged objects.
Should	Respond to *every* object and contact. Even, if an object is not interactive, the experience still provides subtle feedback that acknowledges the object so users don't wonder if the object is broken or malfunctioning.
	Respond *immediately* to the presence of untagged objects. This immediate response blurs the line between the real and the virtual.

	Make feedback whimsical, magical, and not informational by applying effects at the presence of objects, showing that Microsoft Surface can see them even if it cannot identify them.
	Do not rely on blob properties to denote the precise shape or size of physical objects. The shape of all objects on Microsoft Surface is elliptical, and the size denotes only the IR-reflective portions of objects.
Could	Tag all objects in your application. Even if you do not actually need to recognize objects in your application, we recommend that you do not use untagged objects to generate contacts.
	Tag all objects in your environment. Users are likely to begin to place other objects on Microsoft Surface to learn their reaction. By providing many tagged objects, you can enhance the experience.

2.2.4 Create Single-User and Multiuser Experiences

Microsoft Surface is particularly well-geared to multiuser interaction, but you should also consider how your application will be used by a single user, and how you can encourage that user to recruit other users to the experience.

	Create Single-User and Multiuser Experiences
Must	There are no certification requirements that apply specifically to social elements. Instead, consider the more specific issues that are outlined in the following guidelines.
Should	Create an experience that comes alive with several users, so that the experience is more fun or efficient when many hands are working simultaneously.
	Enable a single user to enjoy the experience without requiring other users.
	Enable new users to join in midstream so that newcomers can easily engage with your application without disrupting other users.
	Make sure your application can continue with fewer users, allowing one user to leave without disrupting all others' experience.
Could	Enable users to divide up their tasks and to decide for themselves whether they will be engaged in a shared-display, single-user session, or in a truly multiuser session.

2.2.5 Support Appropriate Levels of Inter-User Task Coupling

At any given time, multiple users that are working around a Microsoft Surface unit might be engaged in multiple levels of *task coupling*. Consider how to best support different levels of coupling in tasks, and how to support varying levels of coupling within the same application. There are three distinctive levels of task coupling:

- *Highly coupled tasks*: Users help each other accomplish the same task. For example, two users touch two portions of the same object to perform a manipulation, or two users look for the same album in a large collection.
- *Lightly coupled tasks*: Two users try to achieve a result that depends on them both, but they are engaged in different tasks to achieve it (sometimes called *divide and conquer*). For example, one user searches for an album in a large collection while the other users searches for album art to apply to it, or the chief of fire and chief of police sit at a Microsoft Surface unit and manage different elements of a crisis.
- *Un-coupled tasks*: Users share the same space, but they are engaged in separate tasks. For example, two users search through the same collection of photographs, but each user is looking for different pictures, or one user is searching for photographs, while the other user is checking their e-mail.

	Support Appropriate Levels of Inter-User Task Coupling
Must	There are no certification requirements that apply specifically to supporting levels of coupling. The same requirements that apply to space sharing (see also "Consider How Multiple Users Share Space") and orientation of content (see also "Provide a 360-degree User Interface") also apply to different levels of coupling.
Should	Support multiple coupling levels by enabling users to perform tasks together to varying degrees. Do not segment the Microsoft Surface unit into areas for particular functions (for example, this side is for performing task A, and the other side is for task B).
	Enable many users to simultaneously use content and controls. Do not block progress by requiring all users to use a common set of controls. Instead, allow users to break up portions of the task by dividing up the controls.
	Do not break direct-touch input when users are performing highly coupled or lightly coupled tasks because direct manipulations create consequential communication[1]. For example, if users are searching through objects by physically moving them, their progress is clear by the speed of their movement and its location on the screen, seen through peripheral vision. Changing this to a virtual device removes this communication.
Could	Provide methods of dividing up a task with various levels of coupling so users can work in parallel. For example, enable users to define interaction areas that they can dedicate to a particular function by specifying what is performed in a particular region of the Microsoft Surface screen.

2.2.6 Consider How Multiple Users Share Space

An application designed to be used by multiple people simultaneously needs to make it easy for two or

[1] *Consequential communication* occurs when the behavior of users that are interacting with the system also provides another user with information about that interaction. This type of community occurs a great deal in Microsoft Surface applications, because other users' hand movements are always within the field of view.

more users to see, reach, and use whatever they need for what each is doing. This includes but goes beyond an application that uses a 360-degree user interface. There can be instances, for example, where an application does not have a 360-degree user interface but in which various controls, screen elements, and content “belong” to a specific user who may be on any side of the unit.

	Consider How Multiple Users Share Space
Must	There are no certification requirements that apply specifically to supporting multiuser space sharing. The same requirements that apply to orientation of content (“Provide a 360-degree User Interface”) also apply to multiple-user issues.
Should	Support consequential communication by making system changes clear to all users. For example, when a person uses two hands to zoom a map, any observer can clearly see how and why the zoom changed.
	Avoid the use of ambiguous audio feedback by making sure that the success or error of a touch is not tied to an audio cue. There is no mechanism to help users distinguish the cause of two simultaneous audio cues.
	Do not provide multiple system modes for input touches. For example, in a GUI application, when a user selects a property to apply to an object, the mouse pointer changes mode (such as turning the mouse pointer into a paint brush). This concept does not work with any multitouch system, including Microsoft Surface; which 1 of the 52 contacts should become a paint brush? This is an important, fundamental difference between single-touch and multitouch systems. This is more problematical in multiuser applications, because one user who puts the system into a specific mode can significantly disrupt all other users.
	Do not attach shared controls to one side of the display, because users will be forced to reach uncomfortably close to another participant to use the control. Instead, enable users to move controls and share them or to dedicate the control to a particular user while they perform some lightly coupled or un-coupled task.
	Communicate ownership through the location of content. If new content is "owned" by a particular user, place it in front of that user. If the group shares ownership, place the content in the center.
Could	Provide modal spaces that allow input to change modes based on the location of the touch. For example, if you want users to be able to paint and annotate an object, provide regions of the screen where they can drag the object and where touches are then mapped to either paint or annotate. Make sure that users can also move these regions to enable users to divide their task.

2.2.7 Provide a 360-Degree User Interface

Users approach a Microsoft Surface unit from all directions. To support multiple users, avoid having content that orients towards one edge of the display. No one likes to read text upside down, and this non-ideal experience creates the impression of a "preferred" side of the Microsoft Surface unit.

The Microsoft Surface SDK includes a **ScatterView** control that enables you to orient content towards any edge of the display. This control is the easiest way to achieve a 360-degree user interface.

	Provide a 360-Degree User Interface
Must	Orient the experience to its users by orienting new content or interface elements towards the same direction as the control (and thus the user) that created it. For example, if a new piece of content is a “sub-experience” of a larger one, and that larger one has had an orientation assigned to it by the user, respect that orientation.
	If an application must face one particular direction, orient it towards the side of the Microsoft Surface unit that includes the access point that launched it, or follow the same orientation as Launcher. However, we strongly recommend that applications do *not* face a particular direction.
Should	Enable users to change the orientation of content themselves, and do not tie content to a particular orientation. Users take advantage of orientation as a communication channel (for example, passing ownership or calling attention to a piece of content).
	Make sure all users can read elements. For example, provide multiple copies of textual elements, and use graphical elements that make sense from all sides.
	Make sure all users can reach and use all elements of an application in a useful way, from all sides of a Microsoft Surface unit.
	Communicate ownership through the orientation of content. If new content faces a particular user, the content is interpreted as being owned by that user. You can mitigate this effect by adding multiple elements simultaneously at random orientations (but ensure that the orientation is intentional and not arbitrary).
Could	Design applications in which both the physical environment and software are designed intentionally to position different users toward specific sides of the Microsoft Surface unit. For example, in a banking environment, a bank employee might always be on one specific side of the Microsoft Surface unit and the customers are always on the opposite side.

2.2.8 Support Using 2D Planar Space

Depending on your application’s scenarios and context, the viewable space might be constrained. In some cases, the canvas is fixed, with a limited content presentation. In other cases, the canvas is flexible, enabling users to zoom in and out.

Use spatial memory in situations where the canvas is larger than what appears on the screen. In either case, backgrounds, objects, and controls must consider the z-axis for their behaviors and movements. For example, the Concierge application includes an infinite canvas when users are navigating the map screen, yet category cards and controls stay within the boundaries of the card screen.

	Support Using 2D Planar Space
Must	Create an environment that is optimized for touch in its layout, feedback, icons/images, and behaviors. Any item that responds to touch must be at least 15 mm in size in all directions, and there must be at least 5 mm between minimally sized touch targets. Position interactive elements so that a user's hand, arm, or input object does not block relevant content that surrounds an interactive element. [Application Certification #03]
Should	Leverage spatial memory by enabling users to change the screen layout themselves, and consistently position content and controls within your application. In situations with large canvases, make sure that the spatial relationship of objects is clear and consistent.
	Consider the meaning of spatial relationships. Geographical and other naturally spatial content lends itself well to spatial relationship. For non-physical information, consider carefully how the spatial relationship between elements is considered and remembered. (For example, an organizational hierarchy's levels are strictly hierarchical, because the physical distance between elements has no meaning. However, viewers tend to associate vertical position with power, so the relative position of two equally "ranked" individuals should be the same, and not necessarily moved up because one person reports to a more senior leader.)
	Do not allow one user to shift the views of all users, unless the task is highly coupled (see "Support Appropriate Levels of Inter-User Task Coupling"). In loosely coupled or uncoupled tasks, users are disrupted if the entire canvas moves because of one user's actions.
Could	Use spatial navigation (flat and wide) in place of hierarchical navigation (menus).
	Make sure that the application does not become too cluttered or too sparse. Enable users to quickly and dynamically repopulate the screen with an optimal information density for the task that they are performing (for example, if users are viewing hierarchical data visualizations, provide preferred views of the data and note important information, such as organizational or educational boundaries).

2.2.9 Adhere to Principles of 3D Space Utilization (the Z-Axis)

Users can clearly see and recognize objects, content, and other elements from a distance. When users move closer, they see more detail, such as additional information, subtle textures, or hints of reflected light. When users interact with interface elements, those elements reveal an even finer level of detail through sound, visual feedback, and movement. For example, icons in Launcher transform into application previews when they are touched, and then they change into the live application when they are touched again. These actions all provide progressively more detail with deeper interactions. As users zoom in closer to objects, the objects should reveal unexpected visual or audible details.

	Adhere to Principles of 3D Space Utilization (the Z-Axis)
Must	For all movable and free-form elements, use visual feedback (depth) to acknowledge objects or controls that users successfully touch by moving the item towards the user along the z-axis. (The exception to this guideline is when the z-axis is already being used for another purpose, or where precise placement is required.) [Application Certification #04]
	Adhere to the standard gesture for moving forward and back in the z-axis (see "Use Manipulation Gestures, Not System Gestures").
Should	Use an appropriate 3D projection[2] for Microsoft Surface. A standard perspective projection does not work because users can approach a Microsoft Surface unit from any side.
	Use 3D space in a semantic way, so that the relative z-axis position of each element has meaning to the user.
	Make the structure of every element feel like it has volume. The experience must feel exploratory and invite users to navigate through the volume as if it is their own world.
Could	Use the zoom gesture to move the view in and out, rather than to change the size of content. The functional difference is that all elements move towards the viewer, rather than a single element growing larger relative to the others.
	Give 3D behaviors to 2D elements, so that, for example, users can turn over flat elements and interact with the other side.
	Remember that the potential volume of interactive space can be larger than what users can view on the Microsoft Surface screen at any given moment. Allow users to understand that volume can be a vast 2D canvas and also a fully 3D volume in which content is located and activities occur.

2.2.10 Make Experiences Natural and Better than Real

Build a super-realistic application by creating rich, detailed representations of familiar real-world behaviors that are augmented with delightful capabilities. These two goals are deeply interconnected because "magical" capabilities are more believable if they emerge out of a real environment. For example, photos in Microsoft Surface that users can easily enlarge cannot be stretched in the real world.

	Make Experiences Natural and Better than Real
Must	Create immediate responses to all user input that will receive a response. Pre-buffer content, provide a transition, or use other mechanisms to make sure that every touch receives an immediate and meaningful response. An application without immediate

[2] A *projection* is the mathematical mechanism by which 3D images are mapped onto a 2D plane, usually in such a way that the images appear to be in 3D.

	responses detracts significantly from the user experience. [Application Certification #05]
	Enable single-finger drag and flick movements on movable content. You must always define a single-finger drag-and-flick gesture to make sure that users can always apply these basic manipulations to all content. [Application Certification #06]
	Enable inertia on objects and content that users can move about the screen. The inertia processor in the Microsoft Surface SDK makes inertia simple to implement, and it contributes significantly to the sense of a natural environment. The processor includes two types of inertia: *realistic* and *goal-oriented.* [Application Certification #07]
	Do not use time-based gestures (such as press-and-hold) on content. Time-based activations introduce mandatory delays for expert users, and they also detract from the sense of a natural environment. [Application Certification #08]
	For content manipulation, work in an unmediated fashion directly with the content. UI controls should not be offered as a direct replacement for manipulations (for example, no "rotate button" in place of the rotate manipulation). [Application Certification #17]
Should	Consistently use transitions and make sure the application does not slow the Microsoft Surface unit until the screen and input display random movements.
	Make the experience feel user-driven by ensuring that each state change is clearly in response to user actions. For example, if a user prefers a particular orientation of content, do not "snap" to that orientation. Instead, use a slowing technique that does not employ a step-function.
	Break from "natural" behavior to match user intent. For example, when a user flicks a **ScatterViewItem** control hard towards the edge of a screen, the control should always bounce back only up to 1 inch, rather than what a real-world, physics model would calculate. This break from "natural" behavior matches the user intent to move the control "out of the way." Similarly, when a user flicks a **SurfaceScrollViewer** or **SurfaceListBox** control, the control always moves one page, rather than a physics-calculated distance.
	Do not innovate for the sake of novelty. All interactions in your application should be based on the foundations of the Microsoft Surface SDK (including both the manipulation and inertia processors) or should be natural extensions of the interactions that your users perform.
Could	Consider what advanced expert functionality you want to enable in addition to natural interactions. Provide a mechanism that extends natural behavior to transition the user from a novice to an expert. For more information, see "Scaffolding".

2.2.11 Create a Sense of Life

The application's interface should always show signs of life and rarely be completely still. Moving and changing elements are pleasantly surprising and unobtrusive. Find the Microsoft Surface equivalent to a person breathing or blinking, or clouds gracefully passing in a summer sky. These things are constantly present, but never distracting. And the application should react instantly to any touch and should always provide something else to touch (but nothing to break the flow of attention).

	Create a Sense of Life
Must	All content that responds to touch must do so immediately and visually. [Application Certification #04] Many of the requirements in "Make Experiences Natural and Better than Real" also apply here.
Should	Always show signs of life, even when the user is not interacting. For example, the Water attract application is constantly in motion, but it is never distracting.
	Make sure the behavior is subtle to avoid being annoying or distracting. Do not cause the application's state to actually change; instead, change only background and graphical elements.
	Give instantaneous responses to every touch. These responses help users understand what is happening in the system and why it is happening. Operations that require time to complete should *not* delay the response of the system.
Could	Distinguish between system states and responses to touches. Even when controls or content are disabled, provide a visual response that indicates their state.

2.2.12 Enable Playful, Pleasurable, and Exploratory Touches

Users enjoy "playing" with multitouch input and seeing responses for that input. This includes both responses that are purely enjoyable and not necessarily functional and responses that are integral to the action or purpose of the application. Microsoft Surface experiences should focus on highly crafted representations of content, motion, and sound to help create fun, pleasurable experiences that include the utmost quality and enjoyment. The interaction should focus on the quality of the journey, not just the completion of tasks. For example, the transition from a stack view of photos into a grid view supports the task of browsing more content, but the transition occurs seamlessly as the container's shape changes and resizes and the content gently moves to indicate scrolling

In addition, design Microsoft Surface experiences with a framework that begins with familiar metaphors and objects and then exposes additional possibilities as the interaction unfolds. Over time, this framework enables users to progressively enjoy more capabilities and draws them deeper into the experience.

	Enable Playful, Pleasurable, and Exploratory Touches
Must	There are no certification requirements that apply specifically to this topic. However, many of the requirements in "Make Experiences Natural and Better than Real" also apply here.
Should	Begin the experience with a familiar environment and behaviors, so users quickly feel comfortable in performing explorations. To create this type of experience, mimic the metaphors of Surface Shell or the natural environment around the Microsoft Surface unit.

	Enable quick discovery of delightful interactions, so users can quickly accomplish simple tasks or simply play with the system. Early success creates familiarity, confidence, and a willingness to explore.
	Perform pleasurable transitions by using animation and other techniques.
	Enable users to explore the interface by providing gentle barriers that constrain them to preferred values of parameters but that do not constrain users to a particular value. For example, if an application produces printed photographs, and if a user is resizing a photo, the application might snap to sizes that do not require interpolation of pixels (such as 50% or 25%). However, users should always be free to access *all* reasonable sizes, including 51% and 49%.
	Make the technical performance flawless. The experience should run flawlessly without any delays or hiccups.
Could	Provide continued delight and discovery over time, in minutes, hours, days, or months. For example, the Water attract application begins with gentle ripples to entice users, responds to every touch to give them success, and ultimately draws their attention to the access points to enable deeper engagement.
	Provide a path to transition novices to experts. If the same user will use your application for an extended period of time, create distinct usage patterns and methods for novices and experts, so experts can interact more efficiently. Enable novices to become experts without instructions so they use the application for the long term.
	Provide your application with a personality and make it a character in the experience. For example, a mechanical application uses graphics that suggest it is being driven by gears and motors.
	Enable users to have fun with one another, so that the application drives social interaction. For example, require multiple users to perform a task together by requiring input from many controls simultaneously.
	Consider all constraints in your application, and visualize them as soft constraints. For example, if a user insists on pushing past the end of a list, allow the list to wrap around to the beginning again, after the user has exerted an assertive effort.

2.2.13 Use Affordances and Constraints to Lead Interaction

Most Microsoft Surface experiences are highly exploratory and contextual to the environment. Experiences should invite users to interact when they are comfortable, instead of telling them what to do. Design subtle affordances that invite users to discover through exploration, and create constraints that prevent users from doing any "wrong" action.

	Use Affordances and Constraints to Lead Interaction
Must	There are no application certification requirements that relate directly to providing affordances and constraints.
Should	Provide affordances that invite users to interact even when input is not required. Affordances encourage exploration and provide opportunities for Scaffolding. For example, enable users to always manipulate objects by using the manipulations processor or the **ScatterView** control.
	Provide constraints that prevent users from doing any "wrong" action. For example, do not allow a user to move content outside the bounds of the display, where they cannot retrieve them.
Could	Develop and apply a detailed visual language that indicates where and how users should touch. For example, open forms invite touch, while closed form discourage it.

2.2.14 Ensure the Experience Is Focused

Microsoft Surface experiences should trim features and focus on a few immensely rich options. Narrow the number of choices to the minimum essentials, and focus on making the choices the most rich and rewarding experience possible.

	Ensure the Experience Is Focused
Must	There are no application certification requirements that relate directly to this topic.
Should	Reduce the number of features in your applications. Additional features add both power and complexity. Instead, provide a premium experience in the primary task that the application offers.
	Make sure that the set of features is focused to the particular task. Many applications provide lots of functionality that enable many separate tasks. Make sure that your application's task is clear and that its features focus on performing that task well.
Could	Reduce the number of available paths and choices, so that the next step and available options are always available to users. Achieve the correct balance between the number of choices and paths to ensure that your application meets the functionality needs of its users. The balance is often apparent only by conducting user testing.

2.2.15 Make Applications Intelligent but Not Presumptuous

Microsoft Surface experiences should anticipate user responses to simplify and create efficiency but should keep users in control. For example, if a user is interacting with an application about a recent trip to Europe, the application might automatically display trip photos. However, the automatic action might be presumptuous, disruptive, or annoying when it is wrong. Instead, provide a choice to display vacation photos, making it easy and quick for the user to complete the anticipated action, but leaving the user in full control.

	Make Applications Intelligent but Not Presumptuous
Must	There are no application certification requirements that relate directly to intelligent application design.
Should	Make obvious next steps available in context of the application's use. Completely open applications that provide all functionality at all times are good. But applications that give affordances for the "right" next step and make those steps available are even better.
	Avoid taking automatic high-cost steps. If your application performs some task automatically, always make the user feel in control. And if the task is the "wrong" action, the cost (in terms of time or other factors) of "undo" must be minimal.
	Keep users in full control at all times by enabling them to lead the experience. Do not ask the user to confirm actions (for example, "Are you sure you want to exit?"), but make sure that every step occurs because of the user's explicit actions.
Could	Make non-textual suggestions to users to help them make informed choices. If your application anticipates two possible paths, demonstrate the consequence of those paths instead of describing them. For example, if there are two preferred fonts, put them at the top of a list that users can select from, and illustrate the user's text in those fonts. However, always keep all font choices available.

2.2.16 Integrate Learning with Doing

A Microsoft Surface application should contain levels of depth that smoothly take users from their first touch to full engagement with the application. If instructions are necessary, integrate them with the natural flow of use, and do not steer attention away from content. If you are trying to reveal the existence of functions, make those functions apparent at the right moment (when they have an effect). If you are teaching techniques, such as gestures, demonstrate those techniques through affordances and constraints that guide the gesture.

	Integrate Learning with Doing
Must	There are no application certification requirements that relate directly to learning integration.
Should	Provide a clear path from novice to expert so users can move from the initial view of the application to where you ultimately want them to go. For example, if the novice users are individuals who are working on highly coupled tasks, and you want to perform different loosely coupled tasks, you should visually divide your application with tools to support each task on either side of the Microsoft Surface.
	Make sure essential features are immediately discoverable, so that users can begin a successful journey without rote learning. For example, if your application is about creating a document, provide a blank document and tools for creating content immediately. Do

	not require the user to access a menu to create the blank document and access the most common tools.
Could	Provide instructions within the flow of the application, instead of requiring users to break concentration and search a Help system.

2.2.17 Use Progressive Disclosure to Reveal Functionality

Because of the spatial capabilities of Microsoft Surface experiences, you can enable users to navigate through the environment and provide on-demand, deeper views of an object (also known as *progressive disclosure*). For example, users can zoom into a photo to reveal deeper content and functionality.

The following guidelines describe how to use progressive disclosure to help users learn in Microsoft Surface applications.

	Use Progressive Disclosure to Reveal Functionality
Must	Show users that unseen content exists and show affordances that guide users to access the unseen content. For example, animate a list of songs when it appears on the Microsoft Surface screen so users see the additional content and have visual indication about how to access it (for example, by moving along the same axis as the animation). [Application Certification #13]
Should	Encourage discovery through exploration, so that further functionality is revealed as users continue through the experience. For example, in a music-browsing application, album covers are **ScatterViewItem** controls, so users can touch them and flip them over to reveal the contents of the album.
	Use consistent interaction metaphors within your application. For example, if you use the flipping technique that is described in the preceding item, make all objects use the flipping technique, providing additional interaction capabilities on the back of the object.
	Hint at deeper possibilities, without taking the focus away from the content. For example, when users first launch the Music application, the albums appear on the display, and a few flip over to demonstrate the functionality.
Could	Provide progressive disclosure for content and interaction paradigms and metaphors. For example, when students learn the piano, they are given songs with a limited set of notes. This set expands, as does the complexity of the piece, as the student becomes more knowledgeable.

2.2.18 Ensure Instant Gratification and a Sense of Success

Users might try something that does not work, but the resulting feedback should help them learn, resolve problems, or encourage them in a correct direction.

	Ensure a Sense of Success
Must	Require explicit and intentional user input to activate destructive functions or to cause larger changes or transitions. This input is especially important for transitions that affect more than one user, and even more so when they are engaged in tasks that are not highly coupled. For example, to launch an application from Launcher, users must touch the application once to see the application preview, and then touch it again to open the application. [Application Certification #14]
	Give every touch an immediate visual response, even if the ultimate consequence of the input takes time to compute. A pre-made response is better than a delayed custom response.
Should	Make sure visual indications of touch are accurate so that the user is never misled to believing something is touchable. For example, disabled buttons must be visually distinct from enabled buttons.
	Make sure feedback contributes to a better understanding of the system and its state. For example, when users touch a **ScatterViewItem** control, it moves to the front, grows, and displays a drop shadow, indicating a change in its position along the z-axis and reinforcing its position and demonstrating that it is on top of the content.
	Put users in control, so that they can always understand the state of the application and how to proceed. Do not provide too many automated actions, and keep controls enabled and logical at all times.
	Understand the technical constraints. Performance is only as good as the slowest element in the experience. Provide interactive content or buffer progress, or use other techniques to make sure that users are always engaged and have a sense of immediate feedback.
Could	Make *all* content touchable, so that some visual response is provided no matter where the user touches on the Microsoft Surface screen.
	Buffer progress, so that results do not arrive in bunches, but in a smooth and consistent fashion. For example, when you are loading photos from a memory card, set an achievable pace and display each photo at that pace, instead of in bunches as the system provides them.
	Clarify errors, so that when the users touch the application, they can distinguish between hardware errors (the system did not detect the touch), state errors (the touch was detected, but the touched item is not in a state that they expected, such as disabled), and semantic errors (the touch was detected, the application is in the state they expected, but the application's response to that touch is not what they expected). You can clarify these errors by providing clear visual feedback with information about *all* of these levels.

2.2.19 Appeal to Multiple Senses

Microsoft Surface applications should appeal to more than one sense at a time. Visuals, sound, motion, and physical interactions help communicate, create moods, convey personality, direct attention, and enchant.

	Appeal to Multiple Senses
Must	There are no application certification requirements that deal directly with appealing to multiple senses.
Should	Include pleasing and appropriate visual designs.
	Include sounds that are appropriate to the application. Be sure to consider the Social principle of Microsoft Surface applications and provide feedback that is unambiguous when users are performing simultaneous tasks. For example, if users are performing many rapid input events in parallel, do not give generic error feedback that might be misinterpreted.
	Include visual cues that lead users to discover new places, things, or effects within the application. These cues provide Scaffolding, enable users to explore, and encourages them to reach further by hinting at elements that they have not yet discovered.
	Provide problems that are not daunting by ensuring they do not require frenetic action, impose time limits, or pose threats to the safety of the content.
	Require decision-making based on a set of rules, a context, and a goal. Ensure that the decision-making moments within the experience are at reasonable chunks of time and levels of expertise for your users.
	Deeply engage users with the application by minimizing the user interface and enabling users to focus entirely on their content.
Could	Combine various physical, visual, and auditory responses to create a single, yet complex, effect. For example, moving a physical object through water causes the water to move and provides sounds that enable users to experience the movement of the object and the water.

2.2.20 Use Continuous Input, Not Discrete Actions

Touch, gesture, and direct manipulation in Microsoft Surface experiences move away from discrete actions toward continuous action. In GUI applications, discrete actions are mostly brief, single-click actions that are performed in a sequence to complete a task. For example, to move an object from one location to another, select the object, select the appropriate command, and then move the object.

In contrast, direct manipulation favors continuous actions. To move an object from one location to another, simply grab it and move it to its new location, as the following illustration shows.

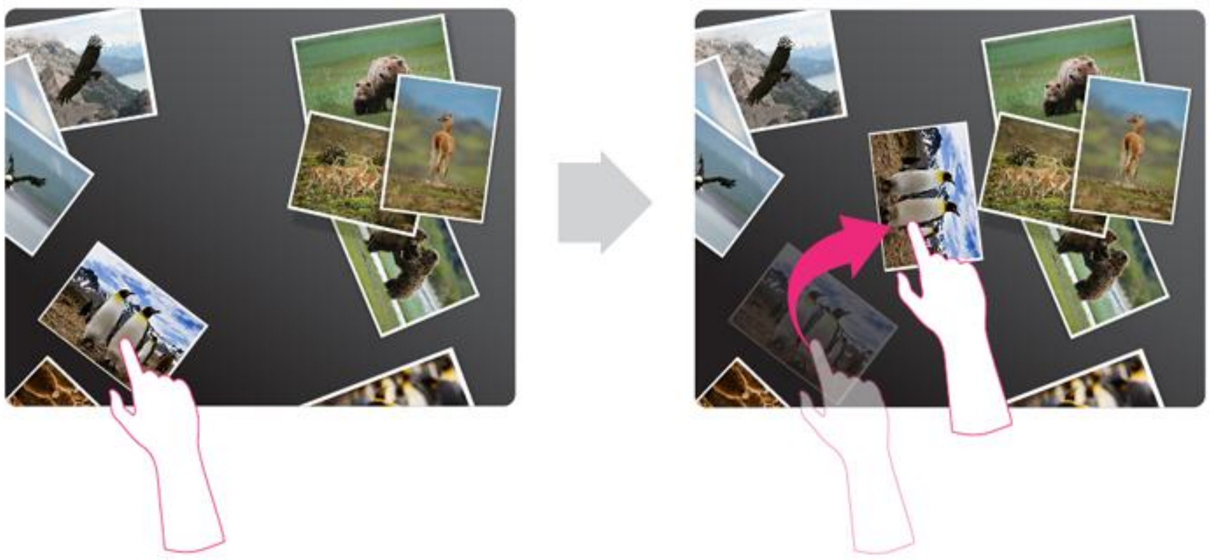


Figure 2-3. A user moves an object from one location to another by using direct-manipulation.

	Use Continuous Input, Not Discrete Actions
Must	Give every touch an immediate visual response, if any response at all, even if the ultimate consequence of the input takes time to compute. A pre-made response is better than a delayed custom response. [Application Certification #04]
	Remember that size matters. In GUI applications, the position of the mouse is represented as a single point on the screen. When fingers and objects are input devices, you must properly size interactive elements to accommodate these input methods, and you must position them so a user's hand, arm, or input object does not block relevant content that surrounds an interactive element. [Application Certification #03]
Should	Place visual responses immediately at the position of the contacts, so that the touch appears like a direct, physical reaction.
	Use many fingers, many users, and many objects by ensuring that your application responds well to many forms of simultaneous input. Use the Social principle and ensure that virtual and physical objects blend seamlessly.
	Use affordances for contact types and methods. Microsoft Surface can distinguish between tags, fingers, and large areas of contact (blobs), but it cannot identify which fingers are on the same hand or user, and it cannot identify different types of blobs (hands versus arms versus untagged objects). Provide visual affordances and constraints to tell users how and where they should contact the Microsoft Surface screen.
	Design for accidental activations, so that users can see and undo actions when they touch

	the screen unintentionally. Accidental activations commonly occur with conversational gestures[3], when draping clothing touches the screen, and when users rest their arms on a Microsoft Surface unit.
Could	Extend direct-manipulation by enabling user input to one area to cause a change in another part of the display. To create this type of manipulation, use Super-realism in the separation between cause and effect, and use the principle of Scaffolding. However, you must also include immediate, in-place feedback, and make clear to *all* users the connection between cause and effect.

2.2.21 Make the Content the Interface

The user interface should have limited items that are not content. Users should directly move objects on the screen. Controls should reveal themselves from content and should be lightweight and relevant for the content. At all times, a Microsoft Surface application should preserve the illusion that users are interacting directly with the content itself. For example, the touch areas of the stack control in the Photos application are at the top of the z-order when the stack is at rest, but the touch areas are partially obscured when users are touching photos within the stack.

	Make the Content the Interface
Must	Do not connect external input devices to a Microsoft Surface unit in such a way that they replace the primary interaction experience. For example, a teacher can use a physical control panel to control display content, but the primary experience for the students must be through touch and multitouch. [Application Certification #15]
Should	Minimize the use of nonsense controls. For example, to zoom an interface, users should use the zoom manipulation, not a button.
	Do not provide multiple system modes for input touches. For example, in a GUI application, when a user selects a property to apply to an object, the mouse pointer changes mode (such as turning the mouse pointer into a paint brush). This concept does not work with any multitouch system, including Microsoft Surface; which 1 of the 52 contacts should become a paint brush? This difference is an important, fundamental difference between single-touch and multitouch systems. This problem is worse in multiuser applications, because one user that puts the system into a mode can significantly disrupt all other users.
Could	Avoid using indirect controls, such as sliders, buttons, and check boxes.
	Demonstrate parameters of the content through physical reactions to the input of the user, using the principle of super realism (see "Super-realism").

[3] A *conversational gesture* is a movement that one user uses to explain or articulate a concept to another user. With Microsoft Surface, this type of gesture occurs most commonly when one user points to an on-screen object, which causes an accidental activation of that object.

2.2.22 Use Manipulation Gestures, Not System Gestures

Gestures refer to physical actions. Many systems use gestures like shortcut keys, so they are independent of the particular physical location where they are performed on the screen. In contrast, Microsoft Surface uses *manipulation gestures*. The distinction between system gestures and manipulation gestures is important:

- Manipulation gestures are physical movements of virtual content within the application.
- System gestures are movements independent of the on-screen content.

Both types of gestures can use identical physical actions, but manipulation gestures are guided and afforded by on-screen graphics, while system gestures are not. If the physical action for users to perform is connected to the on-screen content, you can then provide visual and behavioral affordances and constraints to guide users. The use of manipulations connects well to the principle of Scaffolding.

	Use Manipulation Gestures, Not System Gestures
Must	Do not redefine the following standard manipulation gestures so that different manipulations cause the same behavioral response or the existing manipulations cause different behavioral responses. Use the manipulation processor in the Microsoft Surface SDK to always yield the correct results. [Application Certification #16] • Move: One or more fingers on an item to move or flick it. • Resize out or enlarge: Two or more fingers (points of contact) on an item are dragged apart. Resize in or reduce: Two or more fingers (points of contact) on an item are dragged closer together (that is, toward each other). • Rotate: One finger touches an item and drags it around in a circle so that it rotates and translates; two or more fingers on an item are dragged in opposite directions along an arc (circle); one finger remains stationary acting as a pivot point while others move around it.
Should	Do not replace these manipulation gestures with controls so users can, for example, zoom content by pushing a button.

2.2.23 Properly Integrate with Surface Shell

By properly integrating your application into Surface Shell, the overall context of the Microsoft Surface experience is preserved and ensures a baseline behavior across all applications.

	Properly Integrate with Surface Shell
Must	Provide a video, slide show, or images for the application preview in Launcher. This preview should lead users into your application and teach them how your application behaves. A single static image does not accomplish these goals. [Application Certification #09]
	Include descriptive text with the application preview to give users more information about what they will be interacting with. [Application Certification #10]

	Maintain an attract application within your Microsoft Surface deployment. This application entices users to approach the unit and educates them about multitouch, extending their usage to other applications. Such an application must ensure that every touch receives a response. Multiple touches in the attract application must receive visibly different responses from single touches. Objects (blobs) must receive yet another different response.
Should	Enable access points within your application by ensuring they are visible and visibly distinguishable from the rest of the graphical content.
	Demonstrate successful interactions in your preview movie, helping users know how to perform the first few interactions.
	Transition your application when it loads. Be sure that elements enter the application in a natural transition (see "Make Virtual Objects Behave Like Physical Objects").
	Minimize how many resources your application uses when it is not in focus. Do not design your application so that it continues to require the processor or other system resources when it is not in focus. The Microsoft Surface SDK Help documentation describes how to manage resources by using special .NET event mechanisms.
Could	Follow the color and interaction methods that are provided in the Water attract application and Surface Shell to provide a more seamless experience throughout the system. For example, use water as the underlying metaphor for the application, and extend the use of the linear scroll viewer to drive your experience.

2.2.24 Consider the Physical Environment

The Microsoft Surface experience is influenced by the physical environment around the Microsoft Surface unit, such as the location and audience, the lighting, the objects, and the nature of the experience. For example, cultural differences in audience significantly impact how closely you should place clustered seating, and the seating density also influences how the Microsoft Surface unit is used. Internal décor can also impact how people in a venue might be attracted towards a Microsoft Surface unit.

If a Microsoft Surface experience requires users to place certain objects on a Microsoft Surface screen, the location of those objects and where they are returned when they are not being used can impact how easily users can discover functionality or how users understand the overall goals of the experience.

	Consider the Physical Environment
Must	There are no application certification requirements that relate directly to the external environment.
Should	Consider the physical space where the Microsoft Surface unit will be located. Make sure that your application reflects the environment. For example, if your application appears in lounge environments, match the virtual lighting to the kind of mood that lounge environments likely have.

	Make sure users can access the Microsoft Surface unit from all sides. Never place a Microsoft Surface unit in a location that causes it to directly touch another object, such as another table, wall, or display.
Could	Reflect the *outside* environment in your application, such as the weather, time of day, and season.

3 Visual Design Guidelines

This chapter provides practical design guidelines in key areas to help you design a great Microsoft Surface experience; it is not a recipe book or prescription that outlines exact specifications of Microsoft Surface applications.

Designing Microsoft Surface experiences that incorporate compelling touch solutions requires a different approach to thinking about user interface (UI) and overall graphical screen design than the development of traditional graphical user interface (GUI) products or Web experiences.

3.1 Visual Design Principles

While interaction design establishes defining behaviors, gestures, and responses, it is visual design that brings those elements to life on-screen. Designing visuals for Microsoft Surface applications that are always alive, moving, and responsive to one and multiple touches differs from designing for more traditional computing experiences. The final on-screen visual experience is a major factor in creating user satisfaction, and a key opportunity to cement an emotional bond between the user and Microsoft Surface.

The key to successful visual design for Microsoft Surface experiences is a design language that subtly and subconsciously teaches the user. The user can visually see where to touch, drag, flick, and more without explicit instruction or traditional GUI-based interface elements. This is challenging because while visual design should add beauty and branding to the experience, it should never distract from the content. By focusing on solid design fundamentals that relate specifically to the unique attributes of Microsoft Surface, a designer can create the best Microsoft Surface experiences possible.

Designing for Microsoft Surface, like any other human-computer interface, means creating an extensible design vocabulary with specific attributes, a language of shapes, forms, colors, and controls that help visually guide users through tasks to meet their goals. The following principles inform quality visual design in Microsoft Surface applications and best manifest the Microsoft Surface user-integration goals.

3.1.1 Consistency

Visuals must be consistent to help the users find their through a Microsoft Surface experience There should be a standard set of forms, colors, shapes, textures, and other design elements in an application, or a set of controls that help users orient themselves and anticipate what will happen when they touch something.

3.1.2 Flexibility

Visual design vocabularies should be extensible and flexible. Most Microsoft Surface content and controls can be freely rotated and scaled, so a rigid and inflexible design vocabulary will not be able to accommodate changes in size and shape. A design vocabulary should be able to be extended from one control to another, so that different controls will look related and harmonious.

3.1.3 Premium Quality

Attention to the finest of details can help deliver a top-value “premium” experience. Every piece of content, every control, and every change in application state should be supported by logical, predictable, and beautiful visuals to maintain the users’ context and the continuity experience. Attention to detail does not mean adding superfluous detailing, but rather making sure that those visual details that do exist are highly considered and finely crafted.

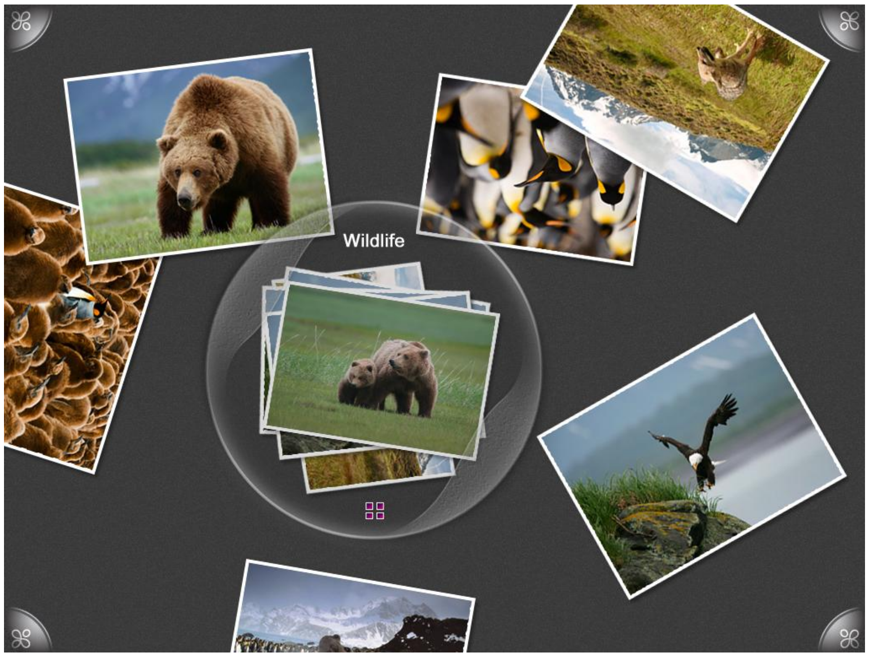


Figure 3-1: A fine attention to detail, even on the simplest and most minimal controls, creates a premium experience with maximum impact

3.1.4 Understatement

Microsoft Surface experiences are typified by a lack of visual noise and clutter. The design must support the content and be beautiful, but without calling attention to itself. Like the best music in motion pictures, the best Microsoft Surface designs are supportive but nearly invisible.

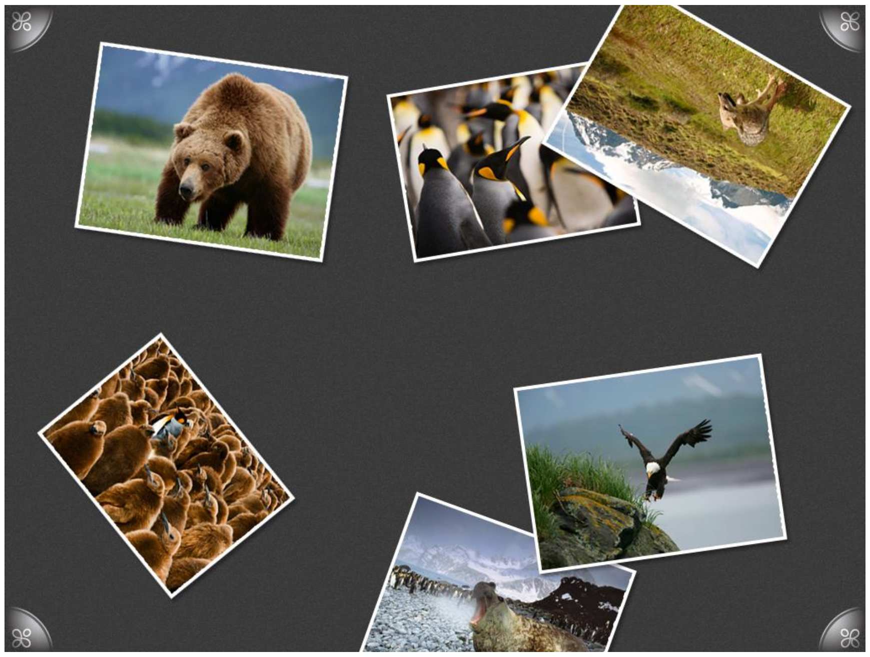


Figure 3-2: These photos scattered on a Microsoft Surface screen encourage direct and immediate interaction; the content is the interface

3.1.5 Minimalism

In a Microsoft Surface experience, content should be the interface itself, so it is important to keep visual elements to a minimum. A "less is more" aesthetic goes a long way towards this goal. Ornamentation and detailing should only support the discovery of on-screen controls and functions, and perhaps some clue as to their function. Eliminating all unnecessary ornamentation, excessive detailing, and needless controls will let the content take center stage at all times. Minimal doesn't mean small, but rather elegant and simple, so be sure to maintain a proper sense of scale to preserve text legibility, the users' context, and their sense of place in the experience on the whole.

3.1.6 Welcoming

Approachable, discoverable, forgiving, and exploratory: these are all attributes of Microsoft Surface experiences that can be heavily influenced by visual design. Colors and forms should be warm and welcoming, and erroneous or mistaken touches should never be met with jarring results or the perception of failure. There should be no fear or exclusivity in a Microsoft Surface experience, but instead provide an environment that is welcoming, inclusive, and rewarding.

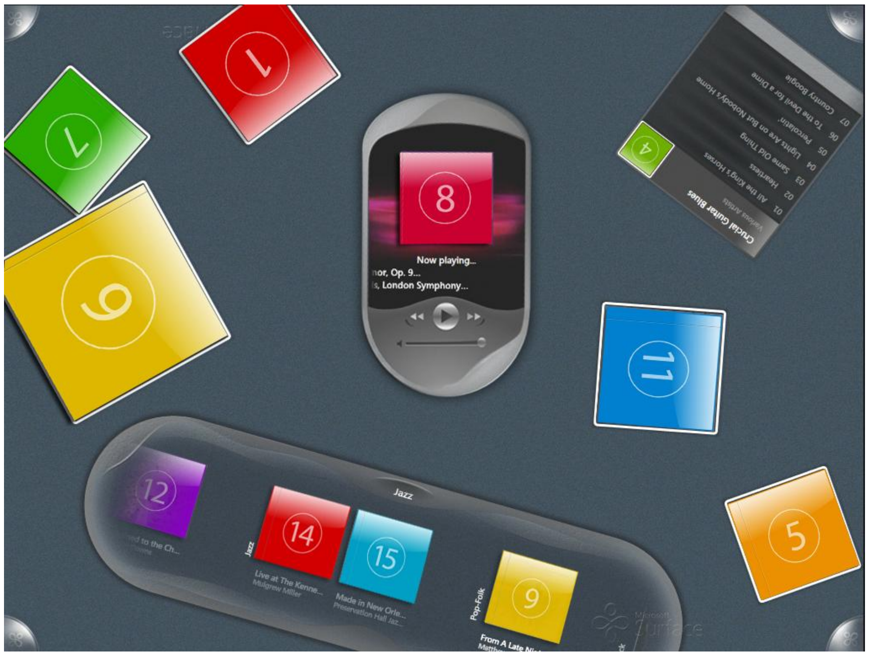


Figure 3-3: The Music application uses CD covers for direct manipulation, and a friendly, approachable, jukebox-style virtual device for playback

3.2 Visual Design Guidelines

The following guidelines provide developers with practical ways to embody the visual design principles discussed above.

Each set of guidelines has two categories. (There are no visual design characteristics that are required by the Microsoft Surface application certification program).

- The ***Should*** category describes guidelines that provide excellent experiences for users and that you can implement at a relatively low cost by using Microsoft Surface tools.
- The ***Could*** category lists guidelines that we recommend so that your application provides a more complete, desirable, and fulfilling user experience. However, these guidelines might also cost more so you should prioritize between them. These guidelines might also apply only to particular application scenarios.

3.2.1 Visual Branding Principles

Extending a brand to the Microsoft Surface experience is a bit different than extending brands to the Web, broadcast, or other media because *the experience itself is the brand.* The elegance and ease of the Microsoft Surface experience should be closely associated with the business and brand that is offering it; it is through having a Microsoft Surface experience that a consumer or user can gain a positive brand impression.

3.2.1.1 Microsoft Branding Requirements

There are certain aspects of the Microsoft Surface experience that should never be modified in order to retain predictability of interactions and user satisfaction. These requirements, as well as guidelines for signage, cobranding, and so on, are fully described in the Microsoft Surface Identity Guidelines[4].

3.2.1.2 Customer Branding Opportunities

If the best Microsoft Surface interfaces are nearly invisible, how are brands best visually represented? The best approach is to tread lightly by being minimal, understated, and welcoming. Users should not feel like they are being exposed to advertising, which is a pitfall of "over-designing" or excessively branding the Microsoft Surface user interface.

Remember that the visual design of Microsoft Surface software is not the only tool available for brand extension. Subtle visual branding mixed with other techniques can have big impacts. How things move through interaction and motion design, can convey key brand descriptors (such as "agile," "reliable," or "stable") better than static visual design. Audio design offers a huge opportunity for adding pleasant surprises and emotional experiences. The environment in which Microsoft Surface exists can also have significant impact, from signage to seating.

	Branding
Should	For information on requirements for maintaining the Microsoft Surface branding elements, refer to the Microsoft Surface Identity Guidelines document.
Could	Establish your brand with minimal and understated techniques. A subtle use of color-on-color with a brand's name in the background, for example, is an effective technique that does not assault the user's senses. Using audio to softly play a recognizable theme when an application starts is another option.

3.2.2 Layout and Orientation

Laying out visual elements on Microsoft Surface poses interesting challenges with the 360-degree nature of the multiuser interface. In most Microsoft Surface applications, any user from any side should be able to read, understand, and interact with any object on the screen at any time.

There is no necessarily true *absolute direction* within the Microsoft Surface user interface; that is, depending on the application, there is no explicit top, bottom, left, or right. Instead, it is best to think in terms of *relative position* to each user, where an object might simply be rotated or placed towards or away from a user.

You should keep these principles in mind in order to make sure that on-screen objects are not only placed and oriented correctly by default, but easily recognizable from any angle. If an object is movable, its form and design should indicate to any user, on any side of Microsoft Surface, that the object can be freely moved and oriented as the user sees fit.

3.2.2.1 Application Orientation

When an application is launched, ensure that its default orientation is beneficial to most users around it.

[4] https://brandtools.partners.extranet.microsoft.com/Corporate/Guidelines/Product+guidelines/Microsoft+Surface+identity+guidelines.htm

There are two ways to do this.

- If an application must be facing in one particular direction, it should orient towards the access point that launched it or follow the same orientation as Launcher. For example, applications that require immediate text entry, such as logging in, will need to be explicitly oriented in order to facilitate easy access to the relevant controls, such as a virtual keyboard.

  The following illustration shows the normal sequence when a person starts to use Microsoft Surface, moving from the attract application, to the Launcher, to a specific application.

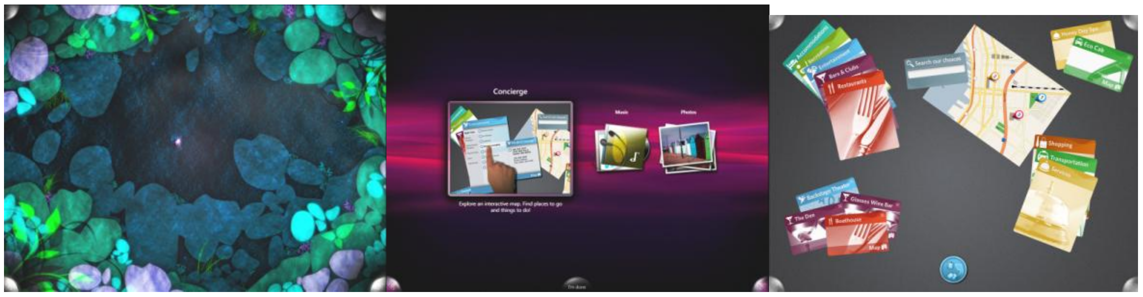


Figure 3-4: Microsoft Surface first orients Launcher based on which access point has been touched (towards either long side of the screen) in the attract application. Then, any application launched will also be oriented in the same way, by default.

- If an application can have its objects freely oriented, rather than the entire on-screen structure required to face one direction, the orientation of those objects should be predetermined in a sensible way, even if the user can change these orientations later. For example, if photos are to be spilled over the Microsoft Surface towards all users, some photos should be oriented towards each side of the screen. This allows anyone on any side of the screen to at least see some photos directly, while placing no restrictions on what they can subsequently do with them.

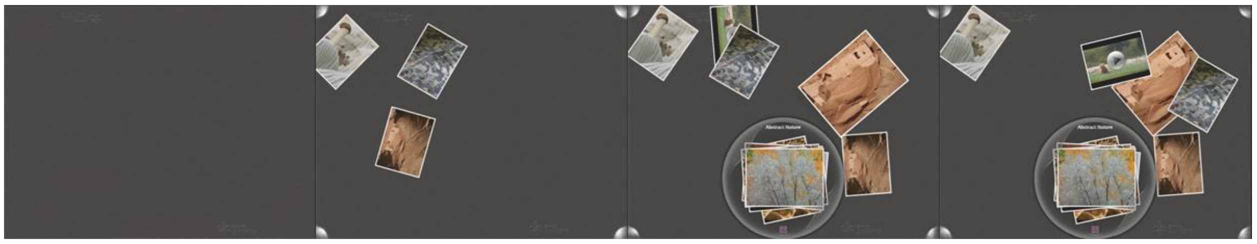


Figure 3-5: While the Photos application seems to arbitrarily distribute photos across the screen, careful inspection reveals that the photos near the edges of the screen are oriented in such a way that any user will at least have some photos that are "right-side up"

3.2.2.2 Gridless Layouts

Most visual designers have learned to create layouts based on grids. The 360-degree nature of Microsoft Surface, however, is conducive to laying out applications without a screen-wide grid system, requiring a fresh perspective on visual layout.

The Microsoft Surface SDK ScatterView control is a natural example of one method for creating a gridless screen-wide layout. It encourages user-driven organization and exploration of content. Acting as an invisible container for objects simply sitting on a tabletop, ScatterView allows for some content to be oriented towards each edge of the screen by default and thus offers natural, direct, and immediate experience (quite unlike productivity-oriented software), all of which encourage curiosity and exploration.

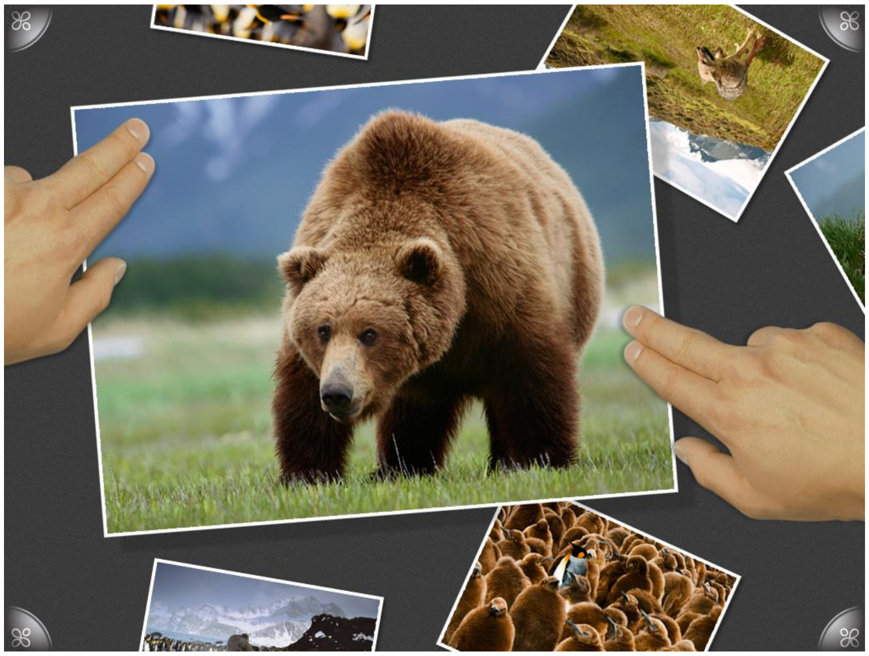


Figure 3-6: The ScatterView control is the perfect example of a Microsoft Surface-appropriate gridless layout scheme that encourages direct and natural manipulation.

Layouts don't need to be gridless all the time. For example, the Photos application makes use of the Scroller control to let the user enforce an order on content so that it may be sorted, filtered, and organized. Therefore, it is possible to use layout grids modally, letting users switch between different visual organization methods as they want.

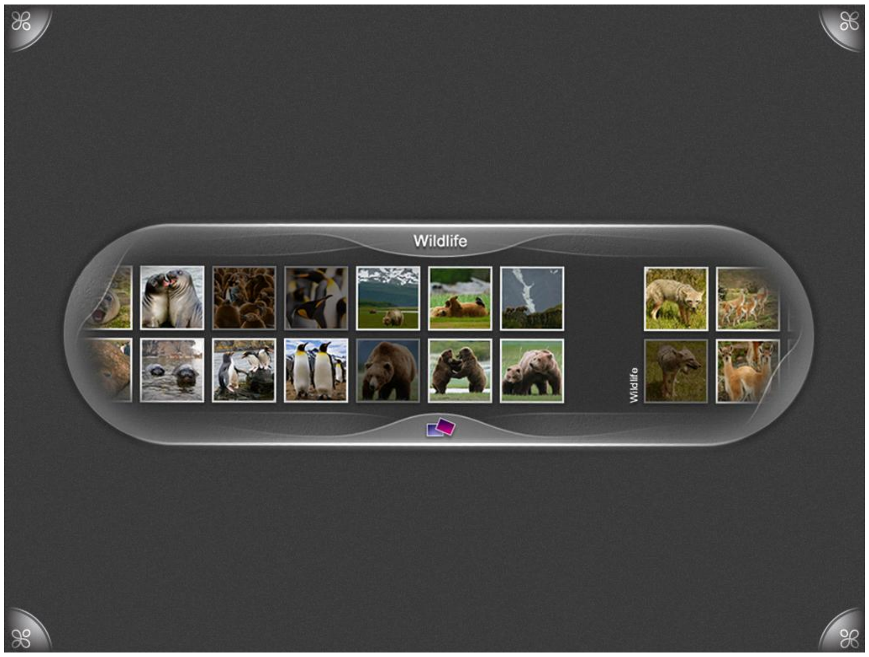


Figure 3-7: The Scroller control lets users enforce order on on-screen objects, allowing for meaningful switching between gridless and grid-based layouts based on the user's needs or context

3.2.2.3 Gridded Layouts

Despite the multiuser, 360-degree Microsoft Surface experience, grid-based layouts still have their place and can be incredibly useful. Gridded layouts are ideal for productivity or focused activities, linear sorting of data, or simply to create a visual rhythm to certain screen states. Layout grids for Microsoft Surface experiences can be thought of as being either *global* (screen-wide) or *local* (within an object, piece of content, or a control).

- *Global grids* determine the arrangement of content and controls screen-wide. While permitting easy scanning and organization, they can force the entire interface to be oriented towards one side of the screen and make it difficult for other users to collaborate or contribute to a Microsoft Surface experience. If you are laying out elements on a global grid in your Microsoft Surface application, give careful consideration to content and control orientation for all users on all sides.

  Global grids are also useful when duplicate controls should be offered to each user, or to each side of the Microsoft Surface screen. This is most important for concurrent, collaborative experiences in which multiple users will need their own sets of controls, to either facilitate real-time interactions or prevent physical interference with multiple users trying to access a single control. For example, if multiple users are playing a card game, each user should have not just his or her own hand of cards, but controls for drawing, discarding, or requesting a new deal.

  However, collaborative action that doesn't need to be concurrent still works best with only one set of controls. For example, users who are simultaneously selecting songs for a playlist in a "jukebox" application would find it confusing to have multiple controls for adding to playlists (as would the application itself) as the playlist attempts to sort out who made which addition in what order.

- *Local grids* are layout systems that apply to specific objects, not to the screen as a whole. They are what give structure to controls and content, making them usable and easy to visually scan. Controls and content tend to be most usable from one orientation, such as a list view control, or any content that is primarily text-based (and needs to be read in a certain orientation). Therefore, it is important ensure that they remain floating and draggable in order to be rotated easily to the right orientation for a particular user.

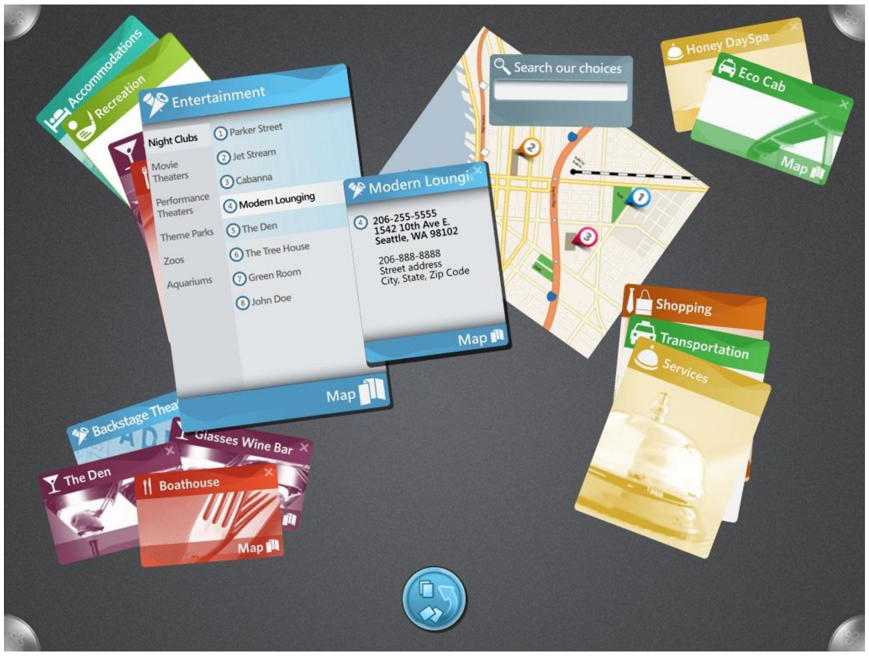


Figure 3-8: While the layout of the Concierge application is free-form and gridless, each element is carefully composed with a local grid to facilitate visual scanning and to create a rhythm of information from object to object

	Choosing Orientation and Layout
Should	Always look for opportunities to move away from global gridded layouts to ensure that applications are equally usable and readable from all sides and by any user.
	For multiuser experiences, plan default object locations and orientations for a 360-degree orientation; each side of the screen should have content facing it.
	Allow users to freely switch from less-structured content viewing (using controls like ScatterView) to highly structured content viewing (using controls, such as Stacks and Scrollers).
	Provide a way for users to reset gridless layouts to their default states. If users lose their sense of place in a free-form, gridless experience, they should have an easy way to regain their bearings and continue.
	Always consider multiple users on all sides of the screen when designing with a global grid system; what's perfect for one user, or one orientation, may make an application difficult for others to use, reducing opportunities for user collaboration.
Could	Use local grid systems within (and between) controls and content so that each on-screen object shares similar visual spacing and rhythms. Ensure such controls have obvious affordances for dragging, rotating, and scaling (as appropriate).
	Look to other media for layout inspirations. Photography's "rule of thirds," for example,

can help add interest and drama to full-screen layouts that are not specific to any particular orientation.

3.2.3 Depth

Microsoft Surface is inherently a spatial experience along the width and height of the screen—the x-axis and y-axis. Using the z-axis, or depth, for content organization and visual prioritization can be incredibly powerful. Depth helps separate foreground elements from the background, giving the user clarity around what to touch. It helps clarify what content or controls have focus or prominence. Depth can also be used to show a sort order or represent a sequence of content pieces. The visual depth of Microsoft Surface experiences should be "shallow" rather than "deep" to help create an airy, natural feeling and easy discoverability. The user shouldn't need to go digging for things, and should be able to reveal what's below quickly and simply.

Depth can be represented using two techniques: two-and-a-half dimensions (2.5D), and three-dimensions (3D).

3.2.3.1 Creating Depth Using 2.5D

2.5D is typified by using techniques that simulate depth without actually requiring 3D geometries. These techniques include scale, drop shadows, transparency, depth cueing, and apparent light sources (see below for a separate discussion of light sources). This can be easily done without specialized 3D tools by using XAML and Microsoft® Windows® Presentation Foundation (WPF) in a high-performance way.

- **Scale** can simulate depth, especially when combined with other techniques. Scaling is an ideal technique to register successful touch input; a slight increase in scale gives the impression of an object being grabbed and moved slightly closer to the user. Large differences in scale between different levels of depths can lead to an appearance of an extremely deep background, so using scale to convey depth is best done in a subtle way.
- **Drop shadows** are an easy and effective way to indicate depth, but it is important not make them too heavy. Dark shadows can obscure content below, and very large offsets can make objects feel disconnected from the Microsoft Surface screen. Drop shadows are ideal for conveying depth relationships between on-screen objects.

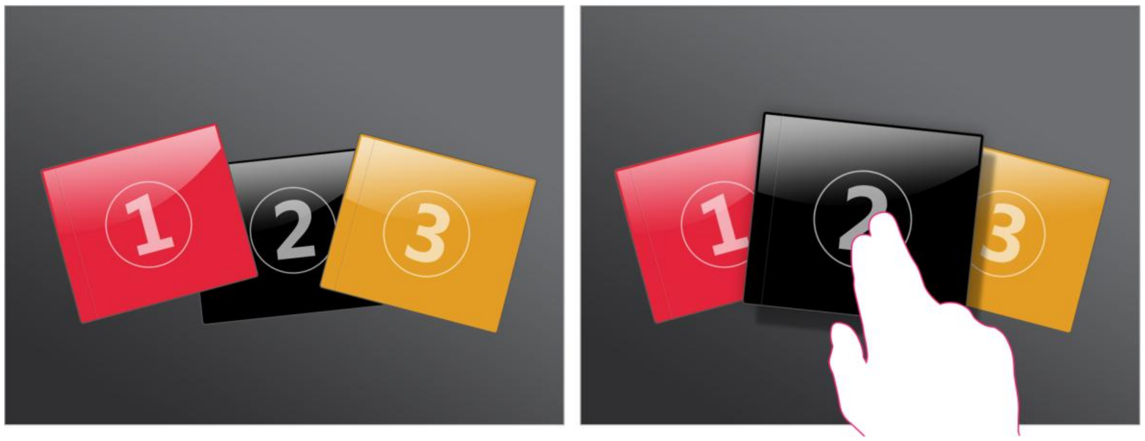


Figure 3-9: Using drop shadows is an easy and effective way to convey depth changes in response to user input

- **Transparency** can show depth by allowing objects on lower levels to show through objects above them. But transparency is computationally expensive to render, so use it judiciously. It is a technique that is good for small numbers of objects on-screen at once.
- **Depth cueing** uses tinting, saturation reduction, and/or contrast reduction in order to simulate distant objects, simulating atmospheric effects like fog. While this can lead to an inappropriately deep sense of distance, using subtle reductions in brightness to convey depth and object focus can be quite effective.

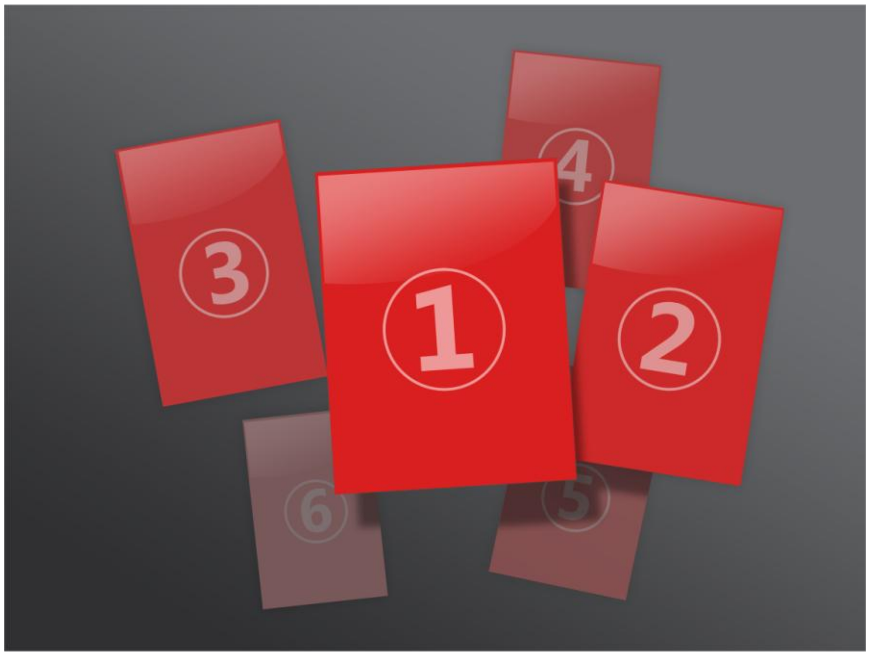


Figure 3-10: This hypothetical example of a book browser shows how depth cueing can be used to separate foreground elements and convey sort order

3.2.3.2 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 tax the Microsoft Surface hardware and impact high performance. True 3D is best created and delivered using Microsoft XNA®, the core Microsoft 3D and gaming engine.

True 3D offers many opportunities for interface innovation, but use caution when designing 3D controls. Controlling a 3D object on a two-dimensional touchable surface can be difficult and confusing to the user. The details of a real 3D object—such as lighting, rendering, and specular highlights—can be overly complex for the needs of most content or data. 3D should usually be avoided for the creation of controls, as any text or labels on the 3D surfaces can become difficult to read. All this complexity can add up to an experience that is so intensely spatial that its 3D nature works against its simplicity and uniqueness.

True 3D space is best reserved for intensely immersive interactions and motion, not calling attention to realistically rendered visual design elements themselves, For example, a continuously zooming interface (maps or Microsoft Virtual Earth™ are excellent examples) use extreme amounts of 3D space for interaction and establishing a sense of spatial memory to facilitate user interaction and preserve user context, but actual 3D-rendered interface elements are practically nonexistent. Consider using depth not to describe on-screen objects, but instead as a virtual space through which to guide the user.

3.2.3.3 Foreground and Background Elements

While subtle depth provides effective cues between foreground objects, such as sort order and touch response, obvious depth provides clear separation between interactive elements and non-interactive spaces, commonly referred to as the "background" of a Microsoft Surface experience.

Backgrounds do not trigger events or actions when pressed, but they should remain aware and responsive to all input, so that the user knows his or her touch has been successfully detected by Microsoft Surface. Backgrounds should truly recede into the background of the user's perception in order to subtly indicate that touches will probably not result in an action. They are best shown as flat planes that are parallel to the Microsoft Surface screen in order to retain cognitive resonance with Microsoft Surface's horizontal tabletop (see "The Microsoft Surface Lighting Paradigm").

3.2.3.4 Microsoft Surface Lighting Paradigm

Lighting is a key way to convey depth between objects and the background, but it is important to understand the theory behind how all objects on the Microsoft Surface platform are intended to be "lit."

The Microsoft Surface experience should be evenly lit from all sides of the screen. The paradigm is not of a light suspended over the tabletop itself, but rather of a huge ring light that surrounds the screen. This is important to emulate if you use drop shadows and other techniques; drop-shadow offsets are the least in the center of the screen, increasing as the object nears an edge, and the offset is always towards the center of the Microsoft Surface screen.

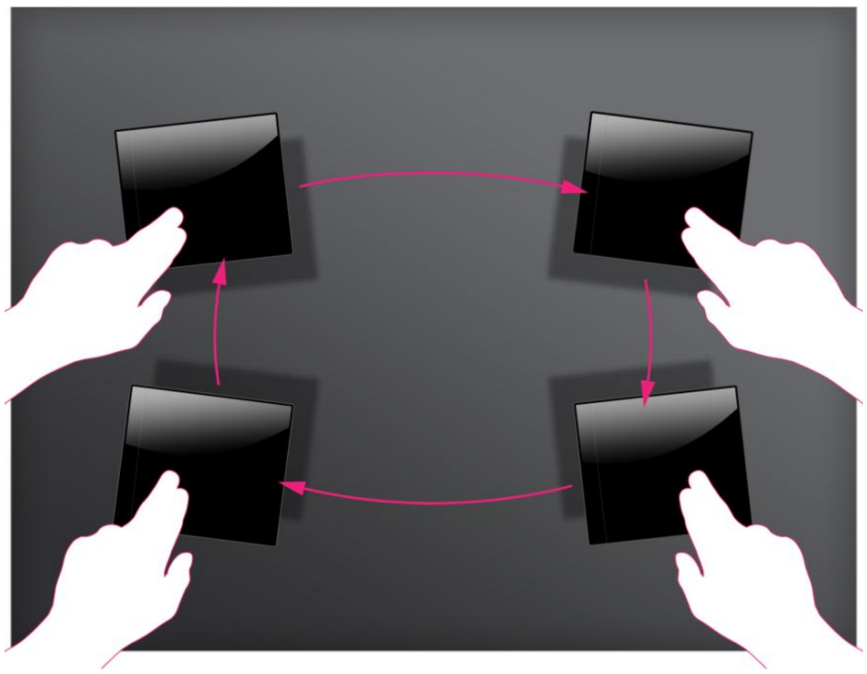


Figure 3-11: Using drop shadows is an easy and effective way to convey depth changes in response to user input

There should never be any simulated light falloff, or darkening, of objects or the background towards the edge of the Microsoft Surface screen. Remember that Microsoft Surface is about superrealism, rather than physical emulation, and physically impossible perfect lighting is one such benefit of this concept.

3.2.3.5 Depth as Affordance

Depth can also be used to delineate zones for certain gestural input, to help a user understand that touching in one region will have a different result than touching in another. For example, an object's header or title bar is likely to be draggable, but a list item within that object would scroll when dragged. Making one portion of a control or object appear to be above or beneath other elements within that object can help make these zones clearer and help users anticipate what results their gestures might produce.

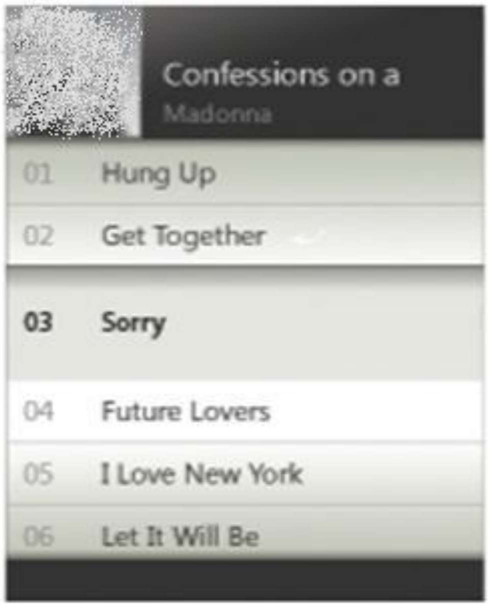


Figure 3-12: This list view is separated from its surroundings using horizontal drop shadows. This helps to convey that the list may be interacted with as a sub-element without affecting the entire object

	Using Depth Techniques: 2.5D, 3D, Backgrounds, and Lighting Effects
Should	Use depth to show priority, order, or focus between objects.
	Keep apparent depth somewhat shallow, not "deep" below the Microsoft Surface display.
	Use depth to acknowledge successfully touched objects or controls. "Float" objects towards the user to register a successful touch, reserving the highest level of depth for those objects actively being touched, so that they float above all other on-screen objects. Many existing Microsoft Surface controls have this behavior built-in.
	With 2.5D, remember to keep depth effects shallow, subtle, and elegant.
	Do not create apparent falloffs in lighting towards the edge of the Microsoft Surface screen, and always orient drop shadows towards the center of the screen, as if it was surrounded by a huge ring light. All on-screen objects, and the background, should appear evenly lit at all times.

	Use depth within controls and content to delineate discrete touchable regions and help users anticipate what results their gestures might produce.
Could	Carefully consider true 3D and its use. Is it appropriate or vital? If so, be innovative with it but take into consideration its programming complexities and runtime performance limitations.
	Be aware of which depth techniques are computationally expensive in order to keep the Microsoft Surface experience highly responsive.
	Use transparency effects sparingly; they can be visually complex and processor-intensive. Applying transparency to solid colors is most effective in reducing visual clutter; transparencies applied to gradients requires more processing power and often make the gradient effect difficult to see. Use extremely subtle (5%-20%) transparency effects to bitmap or raster imagery, in order to reduce visual complexity.

3.2.4 Shape and Form

In order to let content be the interface itself, use shape and form to provide anticipatory clues as to the function of Microsoft Surface objects. Properly defining a shape and form vocabulary directly impacts the quality of a Microsoft Surface experience.

3.2.4.1 Edges and Corners

Volumetric, organic shapes can be friendlier and more approachable than rectilinear forms: this is a key principle in all Microsoft Surface interactions and design goals. Rounded corners and flowing edges are more conducive to interaction; they inherently appear free-floating, draggable, and rotatable, encouraging users to orient controls and content as they see fit.

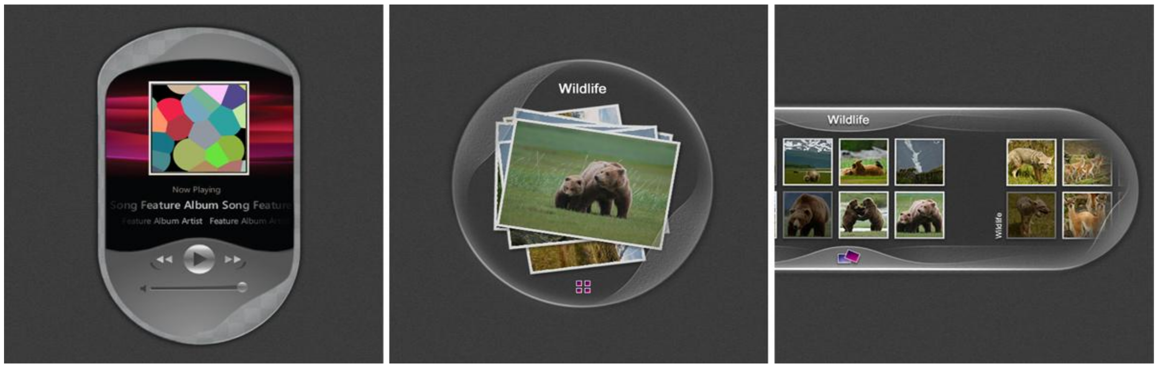

Now Playing
Song Feature Album Song Feature
Feature Album Artist Feature Album
Wildlife
Wildlife
Wildlife

Figure 3-13: Each of the controls above uses rounded, organic forms and subtle shading to appear enticing, approachable, and interactive, while keeping their actual interface elements minimal yet immediately findable

Some objects, are nonetheless still best served by using the sharp edges and rigid lines of their real-world equivalents, such as photographs and CD jewel cases, which are rectilinear in form. In this case, it is appropriate to have their Microsoft Surface equivalents share these visual characteristics, with possibly pleasantly surprising super-realist enhancements and features.

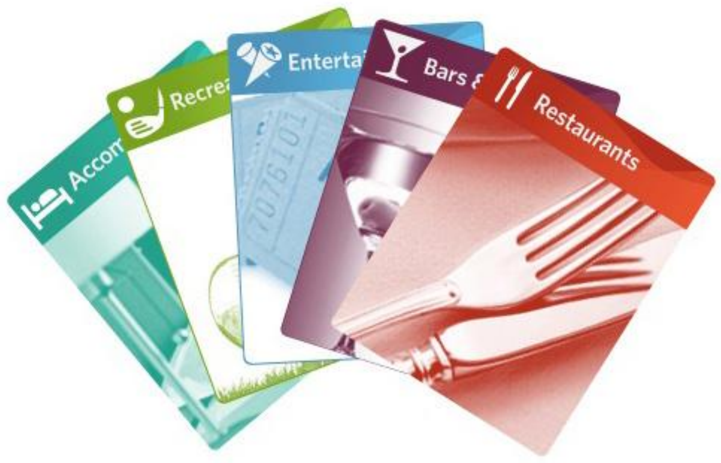


Figure 3-14: Since these interface elements behave and appear like large cards, they appropriately mimic typical cards in real life, with straight edges and bright surfaces

Consistency is vital in determining shape and form. If a control within an application has a rounded header for dragging, other draggable regions or objects should share this same look and feel.

3.2.4.2 Line work

The 360-degree nature of Microsoft Surface applications requires special consideration when it comes to line work. All objects should be able to be freely oriented, rotated, scaled, and moved by the user, which can cause thin lines to appear ragged, soft, or uneven. Thicker lines look better at all possible angles, while thinner ones can develop visual artifacts like "stairstepping" at certain angles. For this reason, line work should generally be at least two (2) pixels in width.

	Using Shape and Form
Should	Rounded forms are generally preferable for approachability, evoking organic shapes, adjusting user preconceptions, and looking more malleable than rectilinear forms.
	Ensure that interactive objects have some sense of volume, relative to naturalistic representation. For example, a photograph should not be volumetric since its real-world counterpart has little to no volume; but shaded corners when flipping the photo over would be totally appropriate. On the other hand, a critical gadget that has no real-world equivalent and thus acts as a virtual device should be made to have some appearance of volume.
	Let the form infer the function. Ensure that interactive objects look like they float, can be dragged or rotated, and so forth. Be consistent in the use of shape and form when developing these form-function relationships; this ensures that user gestures can be consistent, and their results can be predictable and repeatable.
	Rigid or sharp forms are generally discouraged unless the object being designed would have such forms in the real world; but do not overlook the superrealism possibilities of

	stretching beyond what's possible in the real world.
	Keep line work to a two-pixel minimum width for best results when rotating or scaling.

3.2.5 Texture

Texture in software applications and on the web has become highly refined. You see a wide range of effects, such as brushed metal, high-gloss plastic, translucent colored glass, and so forth. While these textures can be functional, most often they are primarily ornamental.

This "conventional" use of textures is not the optimum for Microsoft Surface experiences. While Microsoft Surface is a touch-centric experience, touch on Microsoft Surface does not provide a tactile response that correlates to a specific texture. Overuse of textures, in fact, can distract from the content itself. Textures on Microsoft Surface should have a unique sensibility that is distinctly different than the textures seen on the traditional software.

	Using Texture
Should	Textures, like all other Microsoft Surface elements, can be freely rotated and scaled. It is critical to develop textures that will remain elegant and understated when scaled up dramatically or rotated at odd angles. Fine line work or repeated geometrical patterns often do not maintain their elegant appearances when manipulated in this way. Be sure to test any textures on the Microsoft Surface platform to see how they scale and rotate.
	While textures can act as user affordances, many other visual design techniques can achieve the same goal: negative space, form, shape, color, and more. Explore alternatives to creating repeating textures.
	When using textures, keep them minimal and consistent, and always stay focused on encouraging touch and user exploration.
Could	The clarity and minimalism of buttons can be compromised if they are rendered as 2.5D or 3D elements. This isn't a hard-and-fast rule, but generally the more detailed or highly rendered (that is, a lot of shadowing, gloss, and specularity) a button is, the more overwrought it becomes with unnecessary detail.

3.2.6 Colors

There are no explicit rules around color use in Microsoft Surface applications; designers can and should follow good general color usage practices, as well as those rules imposed by the brands they are handling. There are nonetheless some guidelines that optimize color reproduction for Microsoft Surface's unique screen display.

Microsoft Surface uses a short-throw, rear-projection screen inside the enclosure; light is beamed straight upwards and onto a diffusion screen. This is quite different from a LCD, CRT, or plasma screen. In addition, the Microsoft Surface vision system must be able to "see" through both the screen and the projected light in order to sense touch input and recognize objects. This impacts the final output color gamut and how colors are represented to the user. The gamut of colors that Microsoft Surface can reproduce is narrower than a desktop monitor. Its gamut generally falls somewhere between NTSC broadcast legal colors and offset color printing.

Both brightness and color perception are altered by the Microsoft Surface screen. The brightness gamut is the most noticeable constraint. Pure white has a tendency to "bloom," reducing sharpness and clarity, and pure black renders as a very dark gray. Because the rear projection system is projecting light upward towards the user, large areas of bright or saturated colors can create optical discomfort for users. Cooler colors tend to lack the intensity of warmer hues on the Microsoft Surface screen, coinciding with the ability warm colors have to entice users and put them at ease.

	Using Color
Should	Proof all colors on Microsoft Surface during the design process; iterative proofing is the best way to achieve consistency and optimize brightness and color for the Microsoft Surface screens. Do not rely on a desktop monitor for proofing.
	Design for a brightness range of 90% gray (near white) and 15% gray (near black) to reduce blooming and improve text legibility.
	Bright or saturated tones are best in smaller regions to reduce eye fatigue. Darker, muted backgrounds allow brighter content and controls to take the prominence they need to be usable.
	Warmer colors retain their intensity quite well and reinforce a welcoming and inviting tone better than cooler hues.
	Avoid light from spilling onto the Microsoft Surface display, whether it is from bright ambient light or nearby incidental lighting. Just as a movie screen in a theater, light spill reduces apparent contrast and saturation. In dim environments Microsoft Surface screens have a very pleasing, rich range of tonal reproduction.
	High contrast, such as pure black on pure white, reduces the effectiveness of anti-aliasing (that is, the subtle edge blending of foreground elements onto the background), making text look harsh and introducing "stair-stepping" into object edges when rotated.
	Treat the background as a stage against which all action occurs; it should be the calmest, understated, and de-focused area of the experience.
	Subtle gradients bear careful attention on the Microsoft Surface screen; the reduced color contrast may cause subtle gradients to "flatten" and appear as one continuous tone.
	Be willing to experiment, iterate, and explore as you become familiar with the Microsoft Surface screen display characteristics. Keep an open mind towards taking fresh approaches to a brand's established color palette.

3.2.7 Typography

On-screen legibility on Microsoft Surface is critical. And Microsoft Surface has a number of aspects that make its on-screen reading experience quite unique. With content acting as the interface, and using minimal user interface elements, on-screen text has even more opportunities to act as the interface itself. Content and controls must be equally legible from any angle and from all sides, including upside down, because users may be reading text simultaneously from different sides of the unit.

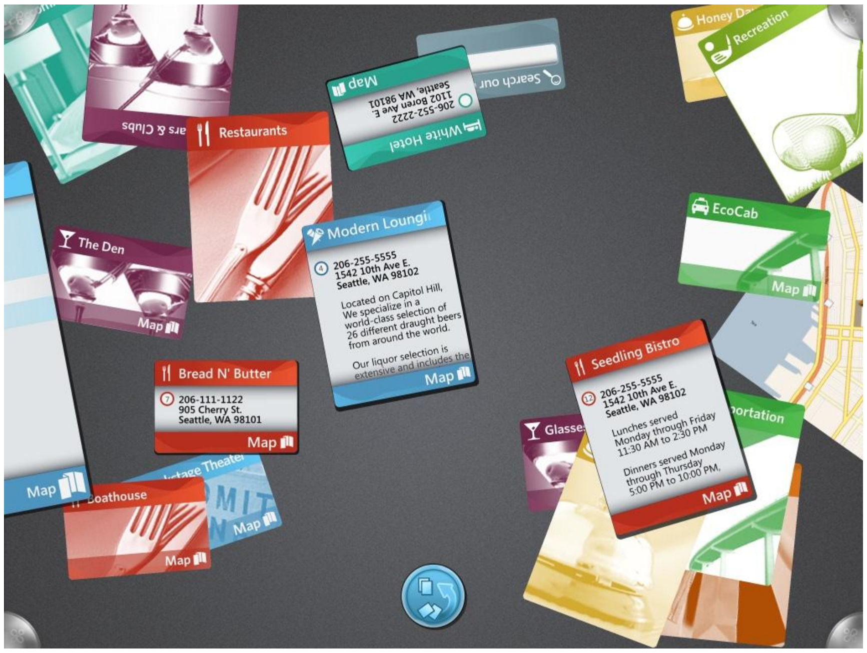


Figure 3-16: Bold, simple typography with classic sans-serif typefaces ensures easy legibility for all users, from all angles, and at most sizes

When choosing a typeface you need to consider free rotation, off-axis legibility, and arbitrary orientation. This generally argues against serif typefaces in favor of sans serif fonts with minimal flourishes, flowing outlines, consistent widths, and generous negative space.

The Microsoft Surface team has had great success using both classic and modern sans serif typefaces, such as Arial, Helvetica, and Microsoft Segoe® that render well on the Microsoft Surface platform.

	Selecting Type Faces
Should	Evaluate text rendering at different angles before and during development.
	Specify all font sizes in pixels, rather than points or em widths. This guarantees that font elements will register to full pixel rows at runtime, increasing legibility on the horizontal axis as well as preserving enough visual data to rotate smoothly and retain legibility.
	Fonts should be at or above 12 pixels in height at all times. No font should ever be typeset below 10 pixels in any circumstance. Scaling fonts up slightly can improve legibility.
	Avoid using all capital letters and small capital letters. Initial capitals are more natural,

	easier to read, and never strike an aggressive tone.
	If serif fonts are an integral part of a visual brand experience, set them at 20 pixels or higher in height for maximum legibility.
	Typesetting in high-contrast reduces legibility. White text on a black background can create bleed, harsh edges, and a poor reading experience.
	Avoid setting text on curved paths possible. If a screen design suggests that text on a curved path is the best approach, be sure to scale the text large enough to be easily legible.
	Remember that most Microsoft Surface objects and controls can be scaled freely by the user. Enforcing minimum scaling limits on objects is sometimes necessary to retain maximum text legibility.
	There is no substitute for testing font rendering on Microsoft Surface itself. Make sure to test for legibility, including when text is rotatable and scalable, if relevant, to get a sense for angles or sizes at which legibility may be compromised.
	A slight, subtle increase in a font's x-height can help open up the letterforms for better reading, especially if you need a slightly condensed appearance for reasons of fit or available space.
Could	Use condensed typefaces with care at small sizes; their lack of interior negative space within each glyph can cause legibility challenges.
	Legibility may be improved by duplicating a piece of text, moving it behind the original text, offsetting it by 1 pixel each horizontally and vertically, and filling it with a dark, neutral tone.
	Always look for opportunities to have on-screen copy held within some other shape to help avoid text floating in space without a container. This also helps to ensure that text will always have a consistent color, background, or tone behind it to improve legibility in all cases.

3.2.8 Iconography

Icons are a staple of the graphical-interface paradigm. They are visual metaphors, compact illustrations that can convey a lot of direct meaning in a small amount of space (for example, a printer icon for printing functionality, a floppy disk for saving, and so on). Once learned, they can be an effective shorthand or replacement for on-screen text.

3.2.8.1 Icons and Objects

In GUI software applications, icons are primarily freestanding graphical shortcuts for functions or menu commands that operate on data (content). And while they can function similarly in Microsoft Surface (such as access points and the I'm Done button), more frequently they are controls in an object or their role is taken over by elements within an object. In other words, it is most appropriate for icons to actually be content, or at least be closely related to content.

For example, rather than designing an icon control that says "Menu" for a restaurant application, it is better to design a virtual equivalent of a menu, which the user is more likely to know how to handle: They can scale it up to look closer, and flick left and right to turn the pages. The concept of superrealism then also applies when they order food: rather than pressing an "add to your order" button, they simply touch the item in the menu itself. In this way, icons and graphical representations of functionality can ensure that Microsoft Surface experiences remain natural, direct, global in interpretation, and contextually relevant.

3.2.8.2 Icons as Affordances

Since icons will always remain useful for inferring functionality to the user, they can still be relevant for some Microsoft Surface experiences. In an effort to create understated and minimal interface controls, the functions of buttons and other controls are not always visually obvious, and it is not always feasible to use text labels on small controls. In such cases, bold and iconic illustrations can be used as effective alternatives to textual button labels.

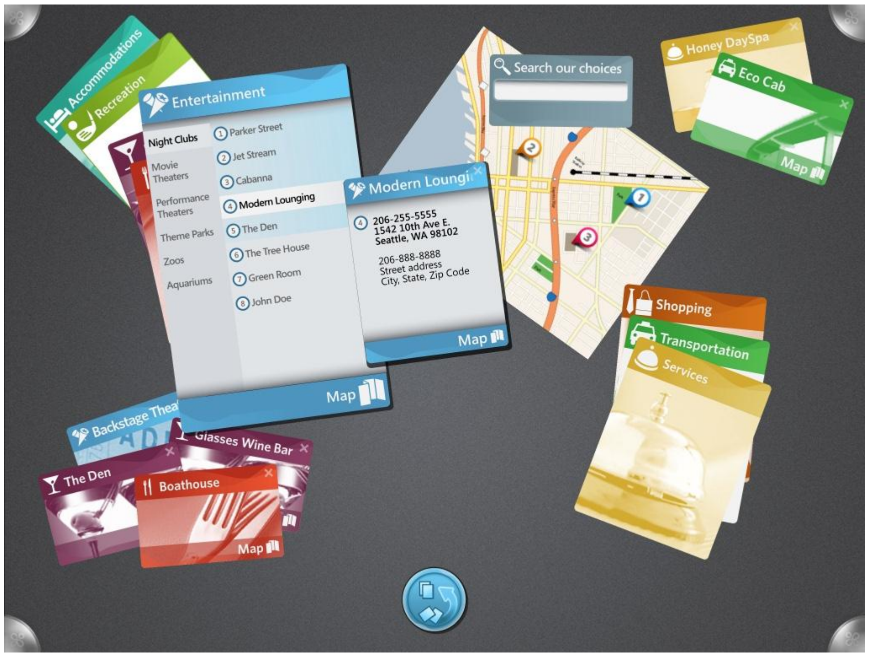


Figure 3-17: The map icon in the top center is simple and bold: It reveals functionality to the user without calling attention to itself. It is also held within a piece of on-screen content, better creating a direct and immersive experience

Icons are tools for visual recognition of application state and functionality, but like any tool, they must be used to solve the right problem. On Microsoft Surface, more direct methods of manipulation are used in the place of most icons. When icons do appear, they must be universal, immediately recognizable, and easily learned.

	Using Icons
Should	In the optimal Microsoft Surface experience content is the interface and the interface is the content. Therefore, icons are best used as affordances, providing hints or clues to users so that they can infer what result a touch or gesture will create. Icons need to encourage exploration and help the interface stay learnable, discoverable, and natural.
	Like typography, icons must be legible from any angle. Do not let icons become complex or overly detailed. Keep them suggestive, simple, and truly "iconic."
	Think globally when designing iconography: avoid culturally specific imagery or references that some users may lack the context to interpret. Icons must be immediately interpreted or, at worst, easily learned and remembered once interacted with.
	Buttons must be sized according to the appropriate input. The touch-based input system on Microsoft Surface requires input devices that are much larger than a traditional on-screen cursor; the size of buttons may need to be larger than is typical for traditional graphic interfaces.
	Icons should be between 5/16 of an inch and 1 inch in diameter in order to be legible, recognizable, and touchable. This limits icon size to between 15 and 43 pixels across, further underscoring the need to design simple and easily recognizable forms.
	Icons should not illustrate, but rather should hint, guide, and infer.

3.2.9 Motion Design

Motion design defines how things move on-screen, and is a critical part of the Microsoft Surface experience. It is motion design that provides the animations and effects that most powerfully convey emotion, action, interaction, system response to touch input, cues and invitations to explore, and screen transitions.

Motion design is never gratuitous in a Microsoft Surface experience; animations always support the content and the experience as a whole. Motion design is created in extensible vocabularies, just like visual design. Transitions are used to provide critical functionality clues and to make sense of application states. Ambient animations help to build both brand and an overall sense of personality for a Microsoft Surface experience. No matter what the technique or effect, motion design should always be consciously used to create experiences that are natural, alluring, and responsive.

3.2.9.1 Designing Motion Design Vocabularies

Motion design should be created as an extensible vocabulary of consistent and shared behaviors, just as a visual design vocabulary is built from a library of color, shape, and typographic rules. A motion design vocabulary should create a sense of rhythm for all movement on-screen.

Rhythm can be created by devising a system of consistent time intervals, which can give a Microsoft Surface experience a palpable rhythm. For example, if objects move in intervals of 150 milliseconds, or half-seconds, pre-animated behaviors will feel consistent and predictable.

Rhythm can also be reinforced by standardizing the apparent weights of objects. On-screen content and controls should feel light, floating, and responsive, with just the slightest sense of weight or inertia. Therefore, all animations should accelerate and decelerate—an animation technique known as *easing*—in order to convey a sense of realism. It is important to remember to use deceleration and the edges of the screen to even dampen velocity, so that no on-screen object can be flicked or thrown off-screen, never to return.

A consistent motion design vocabulary helps to make Microsoft Surface interactions learnable (if animations are tied to functionality), repeatable, and predictable.

3.2.9.2 Importance of the Transition

Motion design governs how all transitions occur. Transitions build context and sense of place for users throughout their entire Microsoft Surface experience. They help users build mental maps of their experiences, show how to use controls by example, and remember where on-screen objects have gone if they are moved.

Transitions are not as important individually as they are collectively; that is, how they all work together and with less dynamic screen states. As a collective animating whole, transitions stitch together discrete moments and actions into a seamless, responsive, natural experience. In this way, transitions are a key building block of an ideal Microsoft Surface experience.

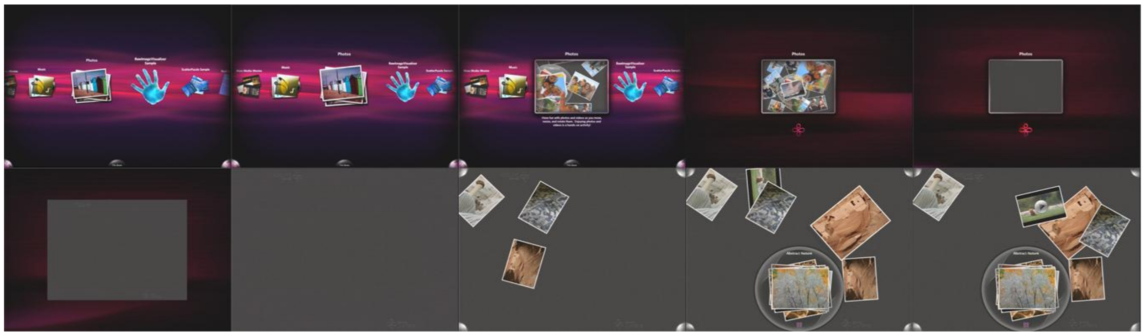


Figure 3-18: This image sequence shows the importance of the transition: seamless, immersive experiences reduce frustration by always maintaining continuity, and helping the user maintain a sense of place and context

3.2.9.3 Latent Learning: Transitions as Affordances

U.S. behavioral psychologist Edward C. Tolman coined the term "latent learning" to describe how learning occurs passively by repeated observation. Any experience with a naturalistic user interface will take some time to learn, like any other human experience, but Microsoft Surface should enable extremely natural, rapid learning and reward exploration with constant enjoyment.

Motion design gives the designer many tools to allow Microsoft Surface users to rapidly learn by passive observation without requiring explicit tutorials or demonstrations. For example, consider an application with a list box control. If the list box suddenly appears on-screen without any transition, it can sometimes appear deceptively static. Are there more options than are shown? Is it scrollable?

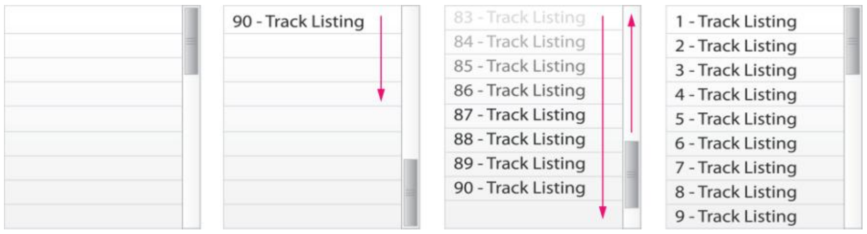


Figure 3-19: Showing a list box's contents slide in when the control is instantiated is a powerful way to convey a lot of information quickly. It suggests that this list is scrollable and that there is more content beyond what can just be seen by default, encouraging user interaction

If the list box appears empty for just a moment, for example, and then its content slides in visually, the user sees this movement and can infer that there are more options than are shown. Likewise, the sliding movement acts as a clue that the content that appears isn't simply static but that it can be manipulated.

This is just one example of how motion design can be used to create transitions that passively and succinctly illustrate how Microsoft Surface content and controls may function.

3.2.9.4 Ambient Animations

Few computing experiences can be said to have much in the way of personality (think of a spreadsheet or word-processing document). But Microsoft Surface experiences are different. They should be immediately engaging and emotional, with a sense that Microsoft Surface has a personality and some sense of awareness.

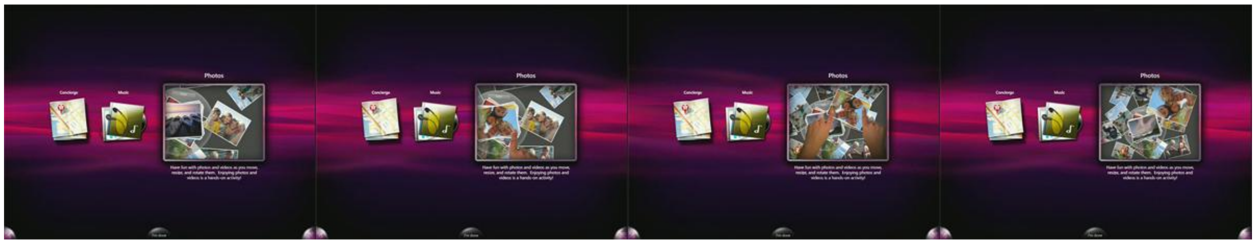


Figure 6.22: Simply having a horizontal ambient animation gives a subtle clue that Launcher is interacted with by horizontal or side-to-side gestures; notice the Application Preview Movie playing, as well

As opposed to entering data in a traditional application, a Microsoft Surface user should see his or her interactions change the state or mood of the Microsoft Surface experience. Subtle ambient movement in the deep background can give a Microsoft Surface experience life, while altering this ambient movement based on user input can give a Microsoft Surface experience a tangible sense of being subtly interactive with and responsive to the user.

For example, a background animation could change in color from neutral to warm based on how many touch inputs are sensed. When several objects are placed on Microsoft Surface and recognized, animations could build visual connections between the objects to illustrate relationships. Visual touch feedback could change shape, color, or rhythm based on how many touch inputs are registered within one object, suggesting that new gestural possibilities exist, and encouraging interaction. Microsoft Surface offers many opportunities for creating ambient animations that help maintain emotional engagement.

3.2.9.5 Effects

Effects can be animated over time to respond to user input; the simplest example is drop shadows (see "Depth") that change their offset and orientation based on where the user moves them. Other effects that can be used at runtime include color and brightness manipulation, blurs, glows, displacement/distortion, and more.

There are two primary challenges in using effects: visual intensity and performance.

Microsoft Surface experiences need to maintain a sense of subtlety and restraint while still feeling aware, responsive, and engaging. A sure way to visually overwhelm the user is to use too many effects, fast movement, and overly energetic transitions. All effects employed must support the overall user experience, and usually the most beautiful effects are the most understated.

Effects are also very processor-intensive, and need to be used sparingly to conserve computing power. If any interaction or animation feels sluggish, the user will sense that Microsoft Surface is being unresponsive, and this will lead to a poor emotional connection with the Microsoft Surface experience. Distortions are especially taxing and should generally be avoided, as should blurring, unless these effects are integral to the content being displayed and the themes being conveyed.

The best way to approach effects is to use them as enhancements, not critical elements. In this way, they can be more easily scaled back, altered, or removed, to improve performance without compromising the visual or motion design of the experience.

	Using Motion, Animations, and Visual Effects
Should	All on-screen objects—indeed, the entire Microsoft Surface experience—should feel smooth, agile, light, and flowing.
	Motion design on Microsoft Surface should never be jarring, sudden, chaotic, or intense. In fact, most of the movement on-screen will be created by the users themselves. Just as in visual design, tasteful restraint and elegant understatement will best serve the Microsoft Surface user experience.
	All objects that can be moved, flicked, and thrown should ease in and out in their movements; this means that their velocity should increase at first and decrease after the user releases it. This guarantees smoother and less jarring movement. Easing in and out need not be symmetrical; that is, the amount of easing in can be different than easing out. Less easing in makes objects seem to have less inertia, and therefore they feel lighter in apparent weight.
	Always establish a motion-design vocabulary around recognizing and rewarding any user input. Even if a touch triggers no specific event, Microsoft Surface must respond visually to ensure the user stays comfortable and knows that Microsoft Surface has recongnized their input. To not respond visually is to infer a failure of the user to interact with Microsoft Surface, which should be avoided at all costs. This should be done for touches, gestures, and even object recognition.
	Never let the Microsoft Surface experience become fully idle. Whether using ambient

	animations *(see above)* or interactive loading states *(see below)*, a Microsoft Surface experience should feel aware and responsive at all times.
	Motion design is a great way to mitigate load times or distract the user during wait states. While all Microsoft Surface experiences should be sufficiently performance-tuned to minimize or eliminate all system delays, sometimes a brief lag is inevitable. This underscores the need for Microsoft Surface experiences to always be reactive, inviting, and fun, even if the user is simply given a fun interactive distraction to distract from a necessary wait. Let the loading process itself be interactive.
	Use the capability of Microsoft Surface to create super-realistic experiences. Just because a sculpted, seemingly solid, volumetric object cannot be rolled or flipped like a piece of paper in the real world does not mean that this is a limitation within a Microsoft Surface experience. Treat superrealism as a license to move beyond literal representations of real-world objects.
	Animations can progressively disclose detail in ways that help preserve user focus and context. A control or content can flow, fold, flip, or transform in reaction to user input to reveal increasing levels of detail or new perspectives on the same data.
Could	Animations can be used to reduce the visual footprint of on-screen controls. For example, a scroll bar may appear only half an inch wide, but increase to be an inch wide when touched, thereby offering both a visual input response and improving the usability of the control.
	Motion design is a fantastic branding opportunity. Never hesitate to interpret brand descriptors or brand attributes into movement; it can invisibly build emotional connections that are inappropriate (or impossible) to convey using static imagery.
	Get inspired. Watch how television commercials use motion to communicate aspects of the brands they are representing. See how the themes of the movie can be expressed in the opening and closing credits for films, especially in the classic work of Saul Bass and the contemporary work of Kyle Cooper and his firm, Imaginary Forces.

4 Interface Text Guidelines

Almost everything that makes Microsoft Surface and the Microsoft Surface experience unique represents a quantum difference from the conventional software experience. Microsoft Surface applications are driven by touch, including and especially multiple touches by multiple users simultaneously. The Microsoft Surface experience is ideally immersive, enchanting, natural, and intuitive: users can see the screen come to life under their fingers and they "know" what to do.

From this perspective, the Microsoft Surface experience should require little user-interface text. But as with all the guidelines in the various areas of this document, applying individual guidelines depends heavily on the type of application you are designing. A commercial application, for example, that describes different cellular telephone plans or shows tourists the points of interest in a city will obviously have to use significant amounts of text. A more strictly entertaining application, on the other hand, such as the Photos application that comes with Microsoft Surface, requires only the most minimal use of text.

The first section of this chapter describes the basic principles that lie behind the use of text and textual elements in Microsoft Surface applications. The second section provides specific guidelines for language and tone. And the third section provides guidelines for using text in what conventionally are specific user-interface components, such as buttons and information messages.

4.1 Language and Text Principles

Whenever a Microsoft Surface application uses text, that text should reinforce the qualities of the experience; it should do its job and be gone. Text should reinforce the Microsoft Surface experience as immersive and natural; if it has to teach the user how to do something, it should do so in a way that makes the learning itself seem intuitive. This is almost always accomplished as much by the style and tone of the text as by its specific content.

When you use text or voice audio, therefore, use them in a way that complements and does not contradict the user experience. Microsoft Surface should not perform like a desktop computer. Many users view Microsoft Surface as a magical device that belongs in science fiction films. Any text and voice audio need to meet such lofty expectations.

4.1.1 Casual, Clear, and Personal Tone

Microsoft Surface is primarily a social experience that is meant to engage and delight, even in a commercial application. To reinforce this, text in Microsoft Surface should speak to users with the same casual and comfortable language people use when speaking with each other.

User-interface text in Microsoft Surface should also be clear and concise. People want to touch Microsoft Surface, so tell them what they need to know quickly and let them get their hands back on the application. Microsoft Surface is unique in that it is one medium in which people do not learn best by reading but rather by doing. Text should point them quickly and let them get to it.

The other side of this principle is that you should avoid computer-based terminology ("computerese") at all costs. A Microsoft Surface experience should be delightful and encourage users to explore and discover. Text designed to help them do this should be friendly as well. Here's a simple example: a text direction for moving photos into a stack should say, "*Slide* the photo into the stack," and not, "*Drag* . . .."

In other words, use everyday words that people use in similar contexts and actions, not words that are associated with functions in GUI software applications.

4.1.2 Using Text Judiciously

Microsoft Surface is fundamentally a visceral, immersive, and touch-based experience, and you should use onscreen text when it is the only way to convey critical information. In default mode, a Microsoft Surface application has access points. Touching an access point takes the user back to the Launcher but leaves the application running. If you want a user to be able to close an application, add an explicit close function. This is what the Launcher does with the "I'm Done" button: it uses text on a button to give a clear choice.

The Microsoft Surface Photos application, however, illustrates another principle. The whole point of the application is that users can move, resize, and play with photos. Yet there is no text anywhere in the application that tells them they can do this and how to do it. A well designed application makes it easy and natural for users to discover themselves how the application works and what they can do with it. Use text if you must, but always consider whether there is another, non-textual way to teach users.

4.1.3 Text as Graphics

On-screen text is a graphical object subject to many of the same principles and guidelines as the integration and design areas. You need to design textual elements as well as write them properly. Font type and size, placement, rotatability, use in applications with a 360-degree UI, and so forth are all things you need to consider. For more information on design considerations with text, see "Typography" in the previous chapter.

4.1.4 Audio

Microsoft Surface comes with built-in stereo speakers and may also be connected to external speakers. And while it may be tempting to use audio to give direction and feedback verbally rather than by displaying text on-screen, audio needs to be used judiciously. On the most basic level, you can never be sure about the volume level of the unit's speakers; how the sound is affected by the physical environment around the unit; or even whether the user is hard of hearing. Although all the text guidelines do not apply directly to voice audio, you should follow all of these principles for both text and voice audio.

4.2 Language and Text Guidelines

Microsoft Surface uses a new language that is casual, comfortable, clear, concise, direct, and personal. It uses real-world, natural terms to go with our natural designs and interactions. When you write text for Microsoft Surface, consider the tone and voice and the word choice.

The following guidelines are designed to provide developers with practical ways to embody the language and textual principles discussed above.

Each set of guidelines has only should recommendations (there are no textual characteristics that are required by the Microsoft Surface application certification program).

4.2.1 Using a Casual Comfortable Tone

Casual language is familiar, informal, conversational, natural, colloquial, and possibly slightly idiomatic. However, casual and comfortable text cannot be too colloquial or idiomatic to the point where it ceases to be clear and concise and becomes costly to localize.
Comfortable language is easily accepted and evokes positive emotions. It is calming and decreases tension that the user might be feeling. Comfortable language tells the user that Microsoft Surface is easy to use.

	Using a Casual Comfortable Tone
Should	Use informal language, as you would when speaking to a friend:
	Good: I'm done *Bad:* I'm outta here *Good:* We can't open Photos *Bad:* The Photos application has failed to launch
	Do not use language that is too casual and relies on slang, colloquialisms, and idiomatic phrases that might be considered silly or are not widely used. Carefully select casual words.
	Good: I'm just a table right now Check back later *Bad:* Out of order – The table has experienced a minor malfunction *Good:* The playlist is full *Bad:* The playlist is maxed out
	Do not apologize. It is hard to maintain a valuable brand and a comfortable experience if Microsoft Surface text blames itself and looks like an unreliable system. If you feel you must be more conciliatory, use "unfortunately" rather than "sorry."
	Good: The video can't play *Bad:* I'm sorry, but we can't play this video *Good:* Unfortunately, we can't read your memory card *Bad:* We're sorry but we can't read your memory card
	Avoid using single words as commands to the user. In addition to being impersonal and computer-like, single-command words are not comfortable or casual. You would never speak that way to a friend. Try as well not to use a single word on a command control, like a button. (Single words are acceptable when you are constrained by space in the user interface.)
	Good: Remove my music *Bad:* Clear

Good:	I’m done
Bad	Reset

Avoid using punctuation with interface text whenever you can. Only use punctuation when you are writing multiple sentences for a description, or when using a hyphen, a dash, or using an apostrophe for casual prose.

Good:	Start a new experience
Bad:	Start a new experience!
Good:	Add songs to the playlist
Bad:	Add songs to the playlist . . .
Good:	Plan A includes 5,000 hours with 400 Anywhere Minutes. Calls are free on weekends and after 7PM on weekday nights. Calls to your Favorite Friends are also free.
Good:	Photos—Explore your memories
Good:	We’re closing everything and starting a new experience

Use capitalization judiciously. Use an initial capital with sentences and phrases. Capitalize brand names. Only capitalize each word in a phrase when the phrase is an important title or label. Never use all capitals or all small capitals.

Good:	Continue my activity
Bad:	Continue My Activity
Good:	Remove a song to add a new one
Bad:	Remove a Song to Add a New One
Good:	Hotels & Motels (subcategory in Concierge)
Good:	Hotels & Motels (subcategory in Concierge)
Good:	Remove my photos
Bad:	REMOVE MY PHOTOS

4.2.2 Using Clear and Concise Language

Clear language is where the meaning is easily understood. Concise language is focused and uses as few words as possible. Keep in mind that Microsoft Surface units are in a commercial environment with very little *dwell time* for each user.

Balancing clarity and conciseness is challenging. Using more words could add to the clarity, but it makes the information less concise. Likewise, using too few words makes the information more concise but less clear. You must satisfy both principles with all Microsoft Surface text and voice audio.

	Using Clear and Concise Language
Should	When appropriate, say what the user would say. For example, if the text is a button or interaction, write exactly what the user wants to say, and say it the way that the user would say it.

Good:	Remove my photos
Bad:	Delete Files
Good:	Continue what I was doing
Bad:	Go back

Do not assume that the user knows what you're talking about. Give enough details to make the meaning explicit. And be specific rather than general.

Good:	Your songs have been removed from the playlist
Bad:	Songs deleted
Good:	View my photos
Bad:	Upload all files on device
Good:	We had to close Photos
Bad:	The application closed unexpectedly

Use as few words as possible.

Good:	Close everything
Bad:	Remove my files, close the applications, and return to the Attract Mode
Good:	Unfortunately, we can't read your memory card
Bad:	We're sorry but we can't read your memory card

4.2.3 Using Text to Add Clarity to the Interaction Design

While good design will make it clear to users what they can touch and do, there are times when text is helpful to create clear interaction design.

	Using Text to Add Clarity
Should	Add text to the interaction design to inform the user what to do.

Good:	Use the keyboard to start a new search
Good:	Drag songs from the albums to the playlist

When the user might be confused or might not understand what a gesture or action accomplished, tell the users that they've completed an interaction. Do not add clarifying text whenever a gesture or action is completed, but only when the user is *prompted* to complete the gesture (like in a game) or when the user might be confused about the interaction.

Good:	We printed your directions
Good:	This song has been added to the playlist

If users' information is kept on the application, such as a credit card number, users are exposed to malicious attacks. If your application removes the users' information, users want to know, so confirm with the users that you removed their personal information.

Good:	Permanently remove all my personal information
Good:	We permanently removed all your personal information

4.2.4 Using a Direct and Personal Approach

Direct personal language simulates a conversation rather than just listing commands and information. Without compromising clarity and comfort, establish a connection with the users by interacting with them in a personal way that you would expect to see in an e-mail message but not in an interface.

Using a Direct and Personal Approach

Should Whenever possible, write from the user's perspectives.

- Do not speak to the users when they are making decisions. Allow the users to speak for themselves. When the user is choosing an option, speak from the user's perspective with "my."
- When the user is informing the application of information, speak from the user's perspective with "I."

Good:	Remove my photos
Bad:	Delete your photos
Good:	I'm not done yet
Bad:	Are you done yet?
Good:	I changed my mind
Bad:	Cancel

When absolutely necessary, speak from the Microsoft Surface unit's perspective and use the term "we." This is most important when a user is waiting, an error occurred, or the user must be told a message, speak directly to the user from the Microsoft Surface unit's perspective.

Good:	We're closing everything and starting a new experience
Bad:	The Microsoft Surface unit is closing everything and starting a new experience
Good:	We can read only one memory card at a time
Bad:	The Photos application can read only one memory card at a time
Good:	We are removing your photos

Bad: All photos are being removed from Microsoft Surface

Do not give Microsoft Surface a "personality." Unless you are developing an application for children, do not turn Microsoft Surface into a specific person with a personality.

Good: We are ready to begin
Bad: I'm glad to see you again. Touch anywhere to begin.

Try to avoid asking questions or giving information and then offering choices. Instead of asking users what they want to do, provide a direct experience by assigning the user's options to the interface. When users are given a button with a clear purpose, they do not have to think about anything except for the purpose.

Good: Close everything
Bad: Are you done?
[Yes] [No]

Good: I'd like to continue
I'd rather go back
Bad: What would you like to do?
{Continue] [Go back]

4.2.5 Avoiding Computer-Based Terminology

Avoid any language and language styles that are common to computers and that are not found in everyday language. Computer-based terminology reminds users of the rigid, impersonal, and overly formal language that they can find in their computer. Use everyday terminology to describe specific actions or activities.

Do not use	Examples
application	While this may be the best word to describe a piece of software in general, but you can avoid it by referring to the name of the application or to the user's Microsoft Surface experience in general *Good:* Close photos *Bad:* Exit the application *Good:* Start a new experience *Bad:* Close all applications
attach	In everyday conversation, people do not say, "Attach this paper to the envelope you're sending." Instead, they say, "Could you put this paper in the envelope?" Instead of "attach," use "add," "include," or a phrase like "put this in" or "stick this on." *Good:* Include my photos *Bad:* Attach all files

cancel	People do not say, “Cancel what you’re doing.” Instead, they say, “Could you stop for a moment and come here, please?” Instead of using "cancel," speak directly to the purpose. *Good:* Remove my photos I’m not done with my photos *Bad:* Delete all files [OK] [Cancel] *Good:* I’d like to continue I’d rather go back *Bad:* What would you like to do? {Continue] [Go back]
click	Since there is no mouse in Microsoft Surface, there is nothing to click or double-click. Use “tap.” *Good:* Tap the button *Bad:* Click the button
confirm	“Confirm” is almost exclusively used in formal transactions but rarely in everyday conversation. People do not use “confirm” when they need to verify information; rather, they just clarify.
	Good: Remove my music *Bad:* Your files will be deleted [Confirm] [Cancel]
data	In conversation people “Tell me about that” or “I’d like some more information”; they do not request data. Instead of "data," use “information.” *Good:* See below for more information *Bad:* Data required
delete	“Delete” is rarely used in conversation. People say “remove,” “get rid of,” and so on. *Good:* Clear the playlist *Bad:* Delete all songs
drag	While “drag” is frequently used in daily conversation, it connotes computer-based language to anyone even remotely acquainted with computers. It would be better to describe a dragging motion with a term like “slide.” *Good:* Slide the photo over here *Bad:* Drag the photo over here
exit	“Exit” is generally reserved for formal and official contexts. People do not say, “Exit the

room when you’re done,” Instead, they say, “Leave the room” or “Go to the waiting room when you’re done.”

- When you are giving a control command, use “I’m done” instead.
- When the term is in a sentence, use “close” instead.
- When the term is used to tell someone to leave, use “go” if possible.

Good: *I’m done*
Bad: *Exit*

Good: Close Concierge
Bad: Exit Concierge

Good: Please go back to the Launcher
Bad: Exit to the Launcher

---

failure
fail

“Failure” had ominous connotations. In conversation people do not say, “There’s a failure in the car,” but rather, “There’s a problem with the car.” Use a word like “problem” rather than the noun “failure.” Use “can’t,” “won’t,” and similar words rather than the verb “fail.”

Good: Unfortunately, we found a problem
Bad: Application failure occurred

Good: Photos wasn’t able to open
Bad: Application Failed to Load

---

file

Rather than the ubiquitous term “file,” refer to the specific type of media or data type (photos, music, videos, postcards, pages, and so on).

Good: Remove all my photos
Bad: Remove all my files

---

load

Nobody would say, “Load your food into the bowl.” Instead, they say something casual like, “Serve yourself” or “Help yourself.” You’re also more likely to hear, “Add paper to the copier,” rather than “Load paper into the copier.” Instead of using "load," use terms like “collect,” “add,” “open,” “search,” or “find.”

Good: We’re opening your profile
Bad: We’re loading your profile

OK

“OK” is quite common in everyday vernacular. But is equally well established in the computer software universe as part of an agree/disagree option pair. In the Microsoft Surface context, therefore, avoid “OK” and speak directly to the purpose. (OK, or “Okay,” is acceptable to use in a sentence if it aligns with the other interface language principles.)

Good: Remove my information
I’m not done yet
Bad: We are about to remove all your information

	[OK] [Cancel]
press-and-hold	Avoid "press-and-hold"; use "tap-and-hold," or "touch-and-hold." *Good:* Touch-and-hold the photo to move it *Bad:* Press-and-hold the photo to move it
reboot	In conversation, "reboot" is used almost exclusively with computers and complex mechanical equipment. Instead of using "reboot" use "restart." *Good:* Restart the game *Bad:* Would you like to reboot the game?
Session	"Session" is commonly used for formal periods or durations. You'll rarely hear, "Are you ending your basketball session?" Instead, people say, "Are you done playing basketball?" • When the term is a command, use "done" instead of "session." • When the term is in a sentence, use "experience" instead of "session. *Good:* I'm done *Bad:* End session *Good:* Start a new experience *Bad:* Start a new session

4.3 Text-Specific Component Guidelines

Even with the new multitouch uniqueness of Microsoft Surface, there are certain instances in which the designer must use text to convey required information. What's more, such text uses conventional components or types, such as text on buttons and error messages. The following guidelines describe how to design and implement text for Microsoft Surface applications. In all instances, the preceding guidelines apply to the specific components.

- Button Text
- Content Titles
- Context Messages
- Interface Descriptions
- Launcher Descriptions
- Giving Users Enough Time to Read

4.3.1 Button Text

Use text on a button when you need to explain the purpose of the button to users beyond what a graphic or icon can clearly convey. Make sure the language is clear, concise, casual, and comfortable. Avoid computer-based commands that you would not find in non-computer interfaces.

Use text when users expect to see it, such as with a check box, in a list of items, or with a radio button.

For difficult-to-explain commands, use design and text on the button to explain the command. As a general guideline, use text as little as possible, but use text to reduce or eliminate any usability issues.

The following examples demonstrate text on buttons and guidelines about how to use text on buttons. This list is limited. You may need other buttons, such as for sending an e-mail, displaying a location on a map, and so on. The same basic guidelines apply to all button text: be specific, personal, and informal.

Button text	Examples
end session	Language and tone should be personal and casual. Use a confirmation after the user's first tap if necessary. *Good:* I'm done *Bad:* End session Reset *Good:* Close everything *Bad:* Confirm
end application	Use this to explicitly close an application, as opposed to having the use touch an access point, which leaves the application running in the background. Specify the application the user is closing. After the user taps the button, use different text to confirm and explain the action if necessary, especially if the user has displayed personal information. *Good:* Close Concierge *Bad:* Exit *Good:* Remove my information and close Concierge *Bad:* Are you sure?
cancel	Consider using a casual phrase, such as "never mind." Be aware of your audience, however, because phrase like "never mind" can be quite idiomatic. You can use "Cancel" if you want to be very clear. *Good:* Never mind *Bad:* Cancel
clear	This is most frequently used to empty a field or other kind of user-supplied information so the user can redo it. Be specific about what the application is clearing. *Good:* Clear the playlist *Bad:* Delete all songs
help	Be specific as to what help is being offered.

	If the context is too general for specificity, use an informal and personal tone. *Good:* Show me how to use gestures *Bad:* Help *Good:* Help me *Bad:* Help
Print	Be specific as to what will be print *Good:* Print my directions *Bad:* Print

4.3.2 Content Titles

In some situations, you might want to add a title to content to help organize or label information. For example, you can place a title at the top of an album that lists songs, or you can place a title at the top of a card that describes a product. However, you should not place a title on a photo, video, or album cover. (The first frame of the video could contain the title.)

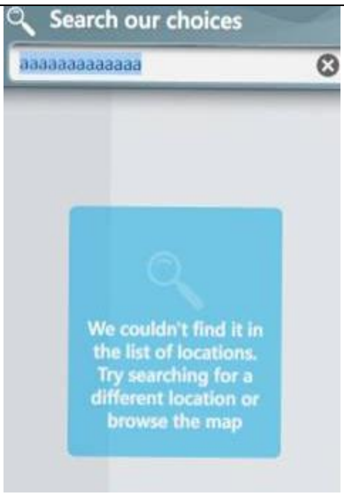
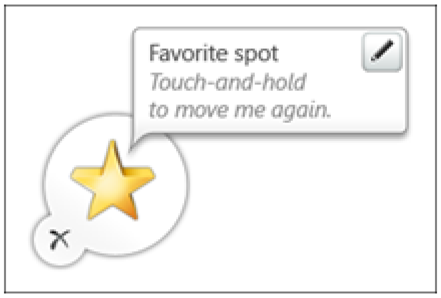
Titles of content are the only text in Microsoft Surface applications where the first letter of each word is capitalized. In all other scenarios, such as buttons, descriptions, and out-of-order text, capitalize only the first letter of the first word.

4.3.3 Context Messages

One way to explain information is through context messages. When a user completes an interaction, your application could open a message that explains the action or what to do next.

The following examples demonstrate some context messages and guidelines about how to use text in context messages.

Context message	Examples
Status	Use text to describe the status and clarify how it happened. You WIN ! Checkmate, the king is in check and has no escape
Search	Use text to describe a result and/or offer some alternate steps to take instead.

	 Search our choices aaaaaaaaaaaa We couldn't find it in the list of locations. Try searching for a different location or browse the map
Conversation bubble	Connect text to a button, icon, or piece of content that the user interacts with. Use text to explain how to interact with an item or what the next step is.  Favorite spot *Touch-and-hold to move me again.*

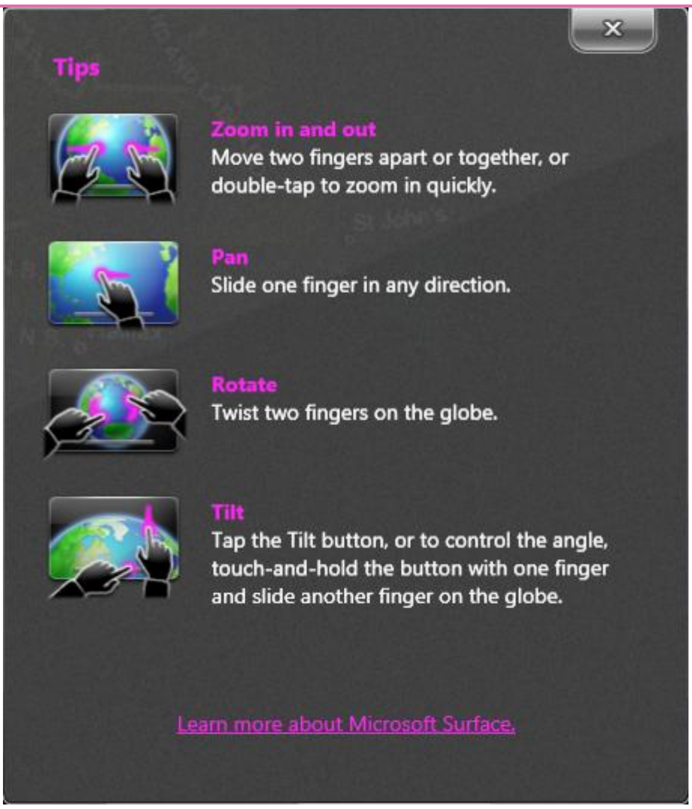
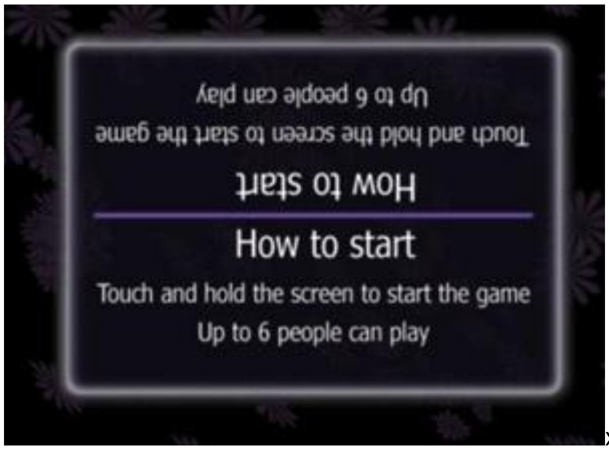
4.3.4 *Interface Descriptions*

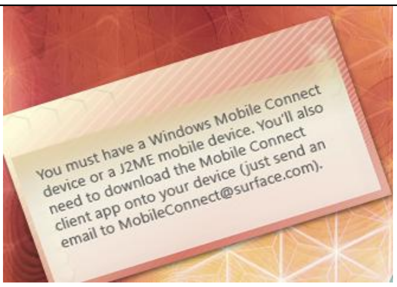
In many situations, you might want to use text to describe any on-screen element, which could be a feature, product, menu item, service, location (such as on a mapping application), news story, gestures, or instructions. You can use descriptions for usability reasons or to simply display information.

The following examples demonstrate some text descriptions and guidelines about how to use text in descriptions of various interface elements.

Note: Not every application will require text descriptions, and most applications will require text descriptions in only one or two areas.

Application	Examples
Gesture tips	Describe the gestures used in your application. Avoid using text only. Consider displaying the information on a **ScatterViewItem** control that users can easily pass around, cover, or move to the side. Include images with your descriptions, or use videos instead. If a gesture involves two fingers, show an image of two hands. Users are less likely to use two hands unless they are taught to do so.

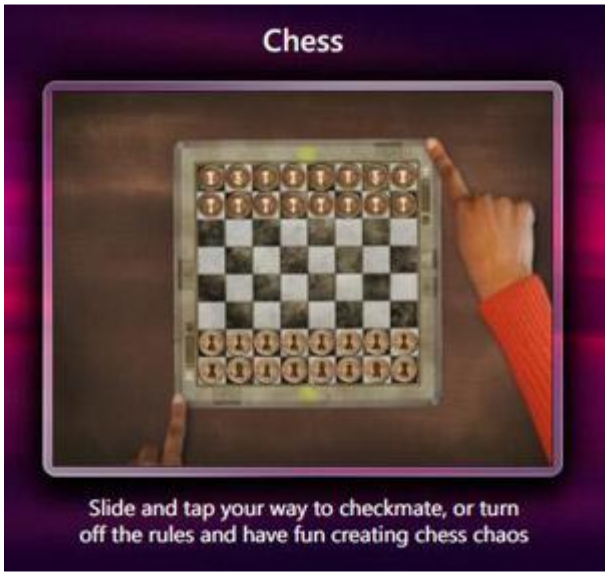
	 Tips **Zoom in and out** Move two fingers apart or together, or double-tap to zoom in quickly. **Pan** Slide one finger in any direction. **Rotate** Twist two fingers on the globe. **Tilt** Tap the Tilt button, or to control the angle, touch-and-hold the button with one finger and slide another finger on the globe. Learn more about Microsoft Surface.
Instructions	Display instructions that a user can read quickly. If your game or application includes special features for multiple people, mention how many people can participate. Consider ways for users on different sides of the unit to get at the directions quickly, easily, and naturally. You could display brief information in two directions simultaneously or put them in a rotatable container.  Up to 6 people can play Touch and hold the screen to start the game How to start How to start Touch and hold the screen to start the game Up to 6 people can play x
Requirements	Display requirements if users need to use specific objects, such as a type of mobile phone, a game piece, a membership card, or a memory card. Display the information on a **ScatterViewItem** control that a user can easily pass around, cover, or move to the side.

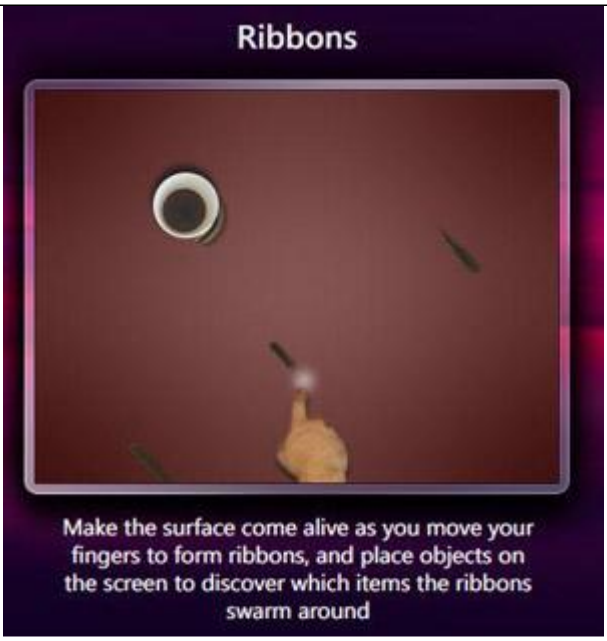
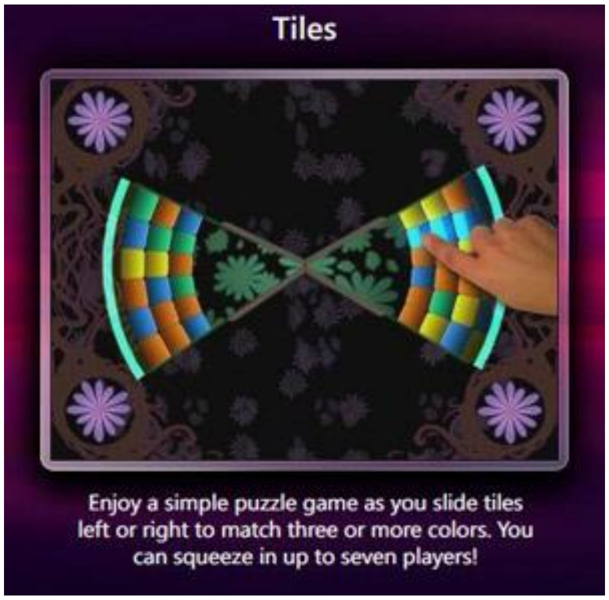
	 You must have a Windows Mobile Connect device or a J2ME mobile device. You'll also need to download the Mobile Connect client app onto your device (just send an email to MobileConnect@surface.com).

4.3.5 Launcher Descriptions

The Launcher menu lists all of the available applications on a Microsoft Surface unit. Each application in Launcher includes a preview image, title, and description. Use the Launcher descriptions to catch a user's attention. Start with a verb and describe the application's best features.

The following examples demonstrate Launcher descriptions and guidelines about how to use text in Launcher descriptions.

Application	Examples
Chess	Describe the specific gestures in the application and the unique features that a user should look for.  Chess Slide and tap your way to checkmate, or turn off the rules and have fun creating chess chaos
Ribbons	Provide brief instructions if the application itself does not include them. If the application includes interactions with physical objects (and if that is not clear from the interface), mention that feature.

	 Ribbons Make the surface come alive as you move your fingers to form ribbons, and place objects on the screen to discover which items the ribbons swarm around
Tiles	If you have two different thoughts to convey, such as what the game is and how many players it supports, use multiple sentences. If your game or application includes special features for multiple people, mention how many people can participate.  Tiles Enjoy a simple puzzle game as you slide tiles left or right to match three or more colors. You can squeeze in up to seven players!

4.3.6 Giving Users Enough Time to Read

For some text in your Microsoft Surface application, you might want the text to appear temporarily and fade away from the screen (for example, notifications, errors, or status information). Any text in Microsoft Microsoft Surface applications should appear long enough for users to read the message before the text disappears from the screen.

Most people can read 5-6 words per second. For every 5 words, keep the message visible for 1 second. For example, if you display "We found a problem; we will now close all the activities and content" (13 words), you should display this text for at least 3 seconds. Also, consider how long it will take for users to focus on your message and become acclimated to it. This might take an additional 1–2 seconds.