Design Principles for Control Software

Contents

Purpose

Software design is an impactful factor in user satisfaction. By understanding user's emotional responses to software design, our designers can steer software design to maximize positive responses and mitigate negative ones. The enclosed principles are intended to serve as guidelines that together drive user-centric software that empower users to reach their goals while simultaneously maximizing pleasure, trust, and happiness, benefitting both users and service owners.

User experience standards set by technology giants are widely adopted and adapted across the software world. In fact, those standards are so successful that the Pew Research Center found that 77% of smart-phone users reported that their device made them "feel happy."[1] Before adopting these principles, it is important to consider the differences between web, mobile, and control software users:

Task	Desktop	Mobile	Control
Primary focus is on the interface	✓	✓	✗
Primary task is content consumption	✓	✓	✗
Primary source of feedback is located on the user interface	✓	✓	✗
Animations drive interaction	✓	✓	✗
Interface is reliably within user's field of vision	✓	✓	✗
First-time users are rare	✓	✓	✗
Outcomes are visible and affect non-users	✗	✗	✓
User is operating device in front of live audience	✗	✗	✓
Primary focus is on tangential devices	✗	✗	✓

This table identifies many ways in which an audiovisual user's needs are different from the needs of someone interacting with desktop or mobile software. Desktop & mobile typically engages the user in a single application or canvas. In contrast, user interactions with control software are typically brief, serve as a means to an end, and in collaboration environments can come with significant presentation anxiety.

Understanding these unique considerations is key to designing effective solutions that optimize positive reactions and perceptions. With these considerations in mind, PepperDash's User Experience (UX) team has developed the following design principles for control software, aiming to maximize efficiency and drive positive user experiences.

[1] US Smartphone Use in 2015. Pew Research Center; April 1, 2015

Golden Rules of UX

The PepperDash Technology User Experience team relies on the following foundational questions when building an interface. All design considerations need to pass three simple tests:

1) Is it simple?
2) Is it intentional?
3) Is it consistent?

Any element that cannot answer yes to these three questions must be justified, or modified. In order to ensure designs meet these rules, all decisions are guided by the following guidelines:

Minimum Pressable Area

In order to ensure that interactive features are easily accessible to all users, PepperDash requires that all interactive elements have dimensions that are at least 1/4 inch or larger – this is referred to as the "minimum pressable area" on an interface. A smaller minimum pressable area may be appropriate for handheld interfaces, because users rely on fine motor control with handheld devices. However, audiovisual interfaces are frequently wall or table mounted, requiring users to rely on gross motor control instead. Proprioceptive interactions are more accurate when relying on fine motor control, therefore target areas for audiovisual interfaces that rely on gross motor controls must be larger than with typical handheld interfaces.

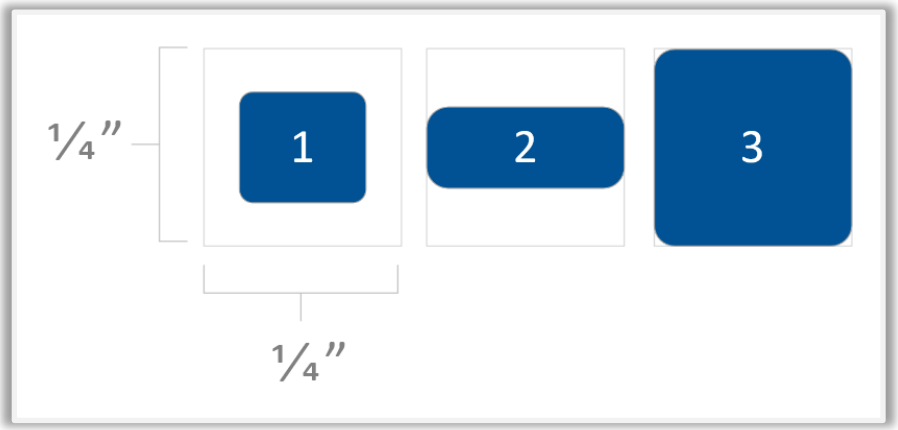

Figure 1
Interactive features may occasionally appear smaller by using graphics with transparent borders, however the pressable target must remain at least ¼ inch.

Because pixel density differs from one device to another, pixels cannot be used as a cross-platform measurement to determine this minimum pressable area. Instead, the display's pixel density must be calculated, and the minimum pressable area is equal to ¼ of the display's pixel density per inch.

Below is an illustration of an interactive element on devices with three different pixel densities.

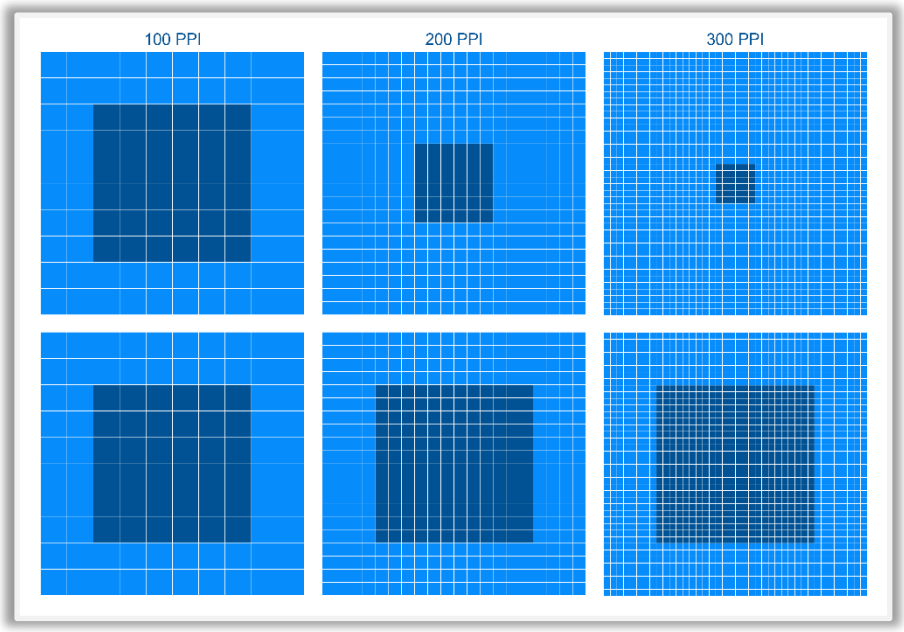

Figure 2
In the top row, elements are the same pixel size. This results in an element that is needlessly large at low pixel density, and unusably small at a high pixel density. In the bottom row, the elements are sized according to the pixel density of the screen. This results in an element that varies in pixel size, but is usable on all screens.

Interface Modality

There are primarily two types of modality in audiovisual user experience design: supplementary, and complimentary. Supplementary modality restricts access to controls or information while complimentary modality provides alternative controls or information. Audiovisual solutions almost always incorporate both types of modality.

Supplementary modality is fundamentally more restrictive than complimentary modality. For example, lighting controls may be available only on a wall switch, while video controls are only available on a touch panel. This is an example of supplemental modality in control. The wall switch is one mode of control, and the touch panel is another mode of control. Supplemental modality can be employed within the interface as well. For example, a simplified set of routing controls might be available to users, while full matrix routing controls are available for technicians. Supplementary modality can be further employed to restrict access to different activities, like when users are restricted from controlling a video call during an active audio call. These are both examples of supplementary modality in behavior. While supplemental modality

is almost always necessary, it should be thoroughly evaluated prior to being employed because of the additional complexity it introduces to the user experience.

Complimentary modality can often offset the detriments of supplemental modality. For example, while a wall switch might be the primary control surface for room lights, complimentary controls might be surfaced on the touch panel as well. This is an example of complimentary modality in control. Or, video call controls might be available during an active phone call. This is an example of complimentary modality in behavior.

Whether modality is complimentary or supplementary, whether modal in control or behavior, the following guidelines should be considered prior to relying on modality:

1) Rely on modality only when required by system design, user privileges, or hardware constraints.
2) Introduce modality only if it improves the user's experience.
3) Choose complimentary modality over supplementary modality unless has a detrimental impact on the user's experience.
4) When supplementary modality in behavior is required, each mode should be visually distinct, and provide an intuitive method to escape that mode.

Hierarchy

A user's experience is most impacted by their needs, and how well a solution meets those needs. Every user has multiple needs when it comes to a controlling an audiovisual solution. One of the most important jobs of user experience design is categorizing these needs into a hierarchy. If this is done successfully, the most important needs are prioritized over secondary needs.

To define an accurate hierarchy of needs, a successful user experience must consider the most commonly performed actions and ensure that these can be quickly and easily reached. Secondary actions should be given lower priority, by locating them on a separate page for example, in order to reduce cognitive load.

An extreme example is seen in interfaces that rely on progressive disclosure, where users are walked through functions with progressive screens. In these types of interfaces, each screen has only a few control options, and the options on the next screen are impacted by the choices on the previous screen. This strategy can often make a complex interaction feel much simpler for uninitiated users by identifying and walking users through their hierarchy of needs on a step-by-step basis.

Forgiveness

A good user experience understands that user error is unavoidable. Because of this, a good user experience will identify potential errors, notify users of such errors, and will provide an easy method of recovery.

An effective user experience should also notify users if their actions have potentially unnoticed, or tangential effects. For example, a privacy button mutes microphones, pauses recordings, and turns cameras off. Because these three effects may not be the primary intention for a user who engages the privacy button, they should be made aware of these effects either before, or after triggering them.

Similarly, if an interaction will result in a critical change or a long wait, users should be given advanced notice of the consequences of that interaction. For example, if pressing a shutdown button will disable the room for several minutes, the user should be offered the opportunity to confirm their interaction prior to triggering the action. This is often achieved via a confirmation or decision dialog. The outcome of such dialogs should never be unclear or difficult to recover from.

Simplicity

Early in the development of touch screen interfaces, skeuomorphic graphical elements were frequently used to beautify an interface, and to orient users to the function of a graphical control. At that time, buttons would appear to physically sink into the screen when pressed. Control knobs, and toggles incorporated three-dimensional shading to teach users that their function was analogous to toggles and knobs in the physical world. As users have become more familiar with touch screen controls, these visual cues are no longer needed, and as a result they have been largely abandoned. In place of drop shadows and inner glows, interface designs now rely on color and saturation to communicate the same thing.

When designing an interface for today's users, the principle "less is more" should be employed as often as possible. Instead of relying on backgrounds, shadows and outlines, rely on white space instead. If white space alone cannot adequately separate controls, then use boundaries or color fields, but keep these elements simple and unobtrusive.

Visual elements should be kept to a minimum when designing an interface. This helps reduce both visual clutter, and cognitive load for users. Before adding any graphical element, consider not only the aesthetic impact of the graphical asset, but also its function. If the element has no function, then its aesthetic value is unlikely to justify its inclusion.

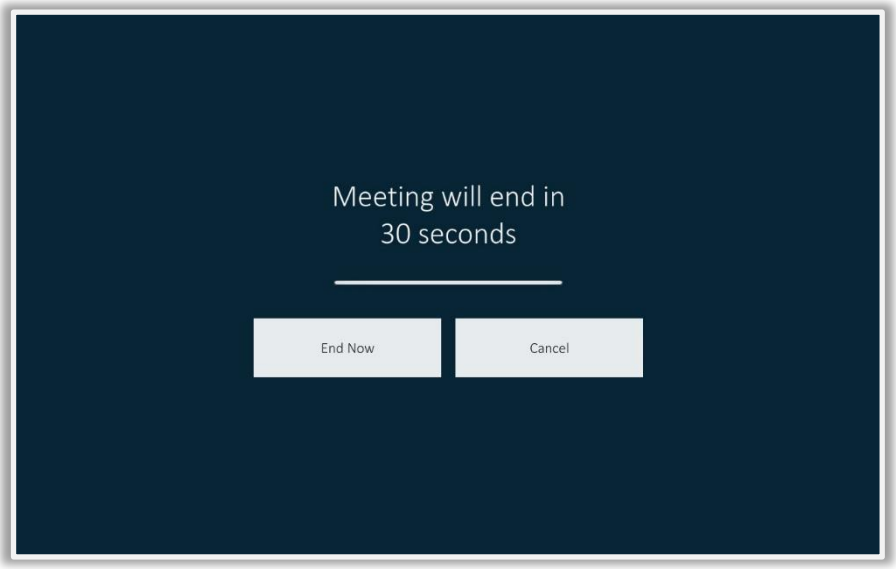

Figure 3
Example of a confirmation dialog

Button States and Modes

Buttons should be designed with four visually distinct states: up, down, active, and disabled. These provide the user with feedback regarding what is unpressed, pressed, selected, and unavailable, respectively. In many situations only up and down states will be needed. However, it is best practice to develop all four button states in order to have consistency, and flexibility for future functionality.

Occasionally, buttons will need to have more than one mode. In each mode, the button can appear in any one of the four states. These modes can be useful to avoid confusion. For example, a green “call” button might transition to a red “end” button when it is active. Or, consider the buttons on a keyboard. Some keyboards do not change the case of the characters on the buttons when caps-lock is enabled. These tend

to disorient users because they need to rely on the state of the shift button when pressing a letter to determine if a press will generate an upper-case, or a lower-case letter.

Figure 4
On this keyboard, the shift button changes states, but the letters on the buttons themselves do not.

Figure 5
Using multi-mode buttons that display the appropriate case resolves the confusion.

Feedback

Provide feedback to the user when they have completed an action. This can be a pressed state on a button, a spinner that lets them know that the system is working, or a status message that provides information regarding what has happened.

Never let the user wait for more than a second without a response from the system.

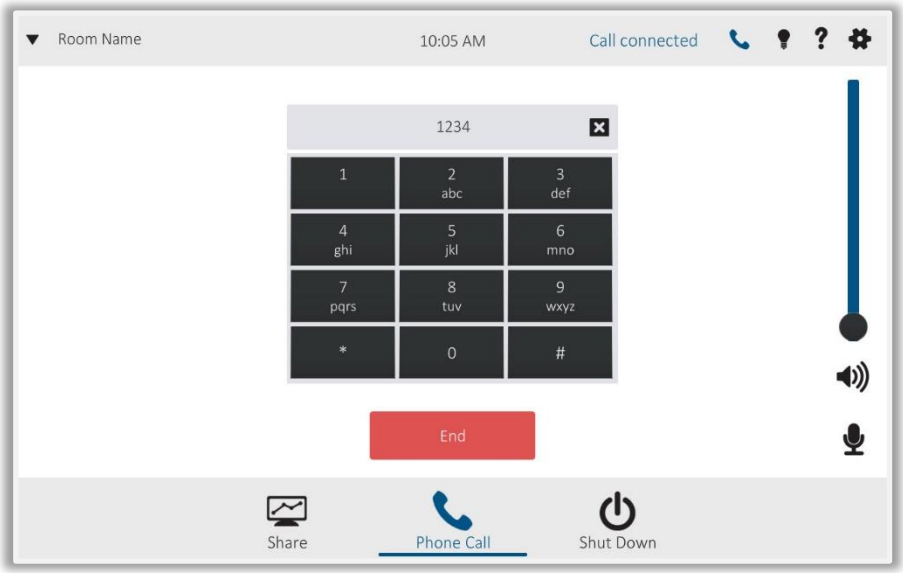

Figure 6
Note the status update in the header indicating that a call is connected. This feedback will persist even when users transition to other activities.

ADA Compliance

Differentiate elements with something besides color (such as text or shape) where possible to enable those with colorblindness to distinguish between them.

You can use online colorblindness tools to see what the design looks like to people with different color deficiencies.

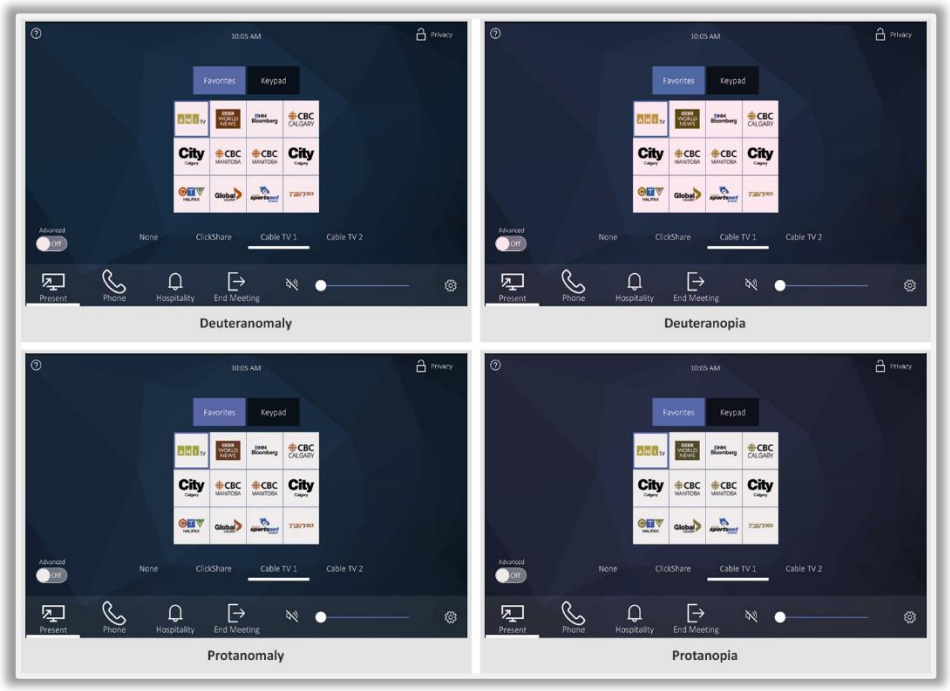

Figure 7
The four most common versions of color blindness have been evaluated here.

Only rely on color to distinguish elements when a secondary means is readily available (such as text labels on the color buttons above).

Special care should be taken when designing an active state. The active state should be visually distinct from the inactive state for all types of colorblindness. This can be achieved through styling (e.g. a border or a skeuomorphic design) or by choosing different values (lightness/darkness).

Language

Language is an integral part of the user experience. Care should be taken to be as clear and concise as possible.

Descriptive Text

Include instructions for more advanced input methods. If an action requires anything beyond a simple button press (holding a button, dragging and dropping, multi-touch gestures) that action should be explained in a nearby text field. An exception would be a "soft lock" where functionality is intentionally hidden behind a press-and-hold.

Don’t assume that the user is familiar with anything beyond button presses and avoid using descriptive language unless necessary.

Simplicity is important: be as concise as possible when writing. Review instructional text and ask yourself if any unnecessary language can be removed. Generally, the less wordy option is preferable: “End meeting?” instead of “Are you sure you want to end the meeting?”

Capitalization

Make sure to use consistent title case and sentence case throughout the entire UI. Titles, labels and buttons should use title case (every word begins with a capital letter). Descriptive or instructional text should use sentence case (only the first letter capitalized).

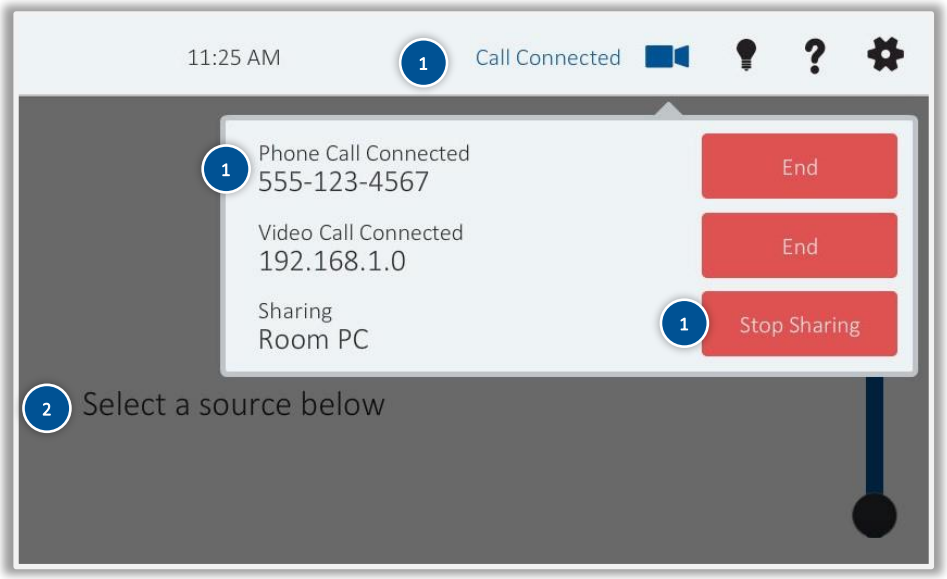

Figure 8
1) Primary text and buttons are in title case; 2) Instructional text is in sentence case.

Punctuation

Punctuation is only necessary when there are multiple sentences of descriptive text placed together, or to indicate an alert (!) or question (?). If a sentence requires a comma, punctuation should be placed at the end. Examples of acceptable sentences include:

"Are you sure you want to power off the room?"
"Select a source"
"Mics, projectors, and volume are muted."

Typefaces and Fonts

Both the number of typefaces and the fonts within those typefaces should be limited.

The need for more than one typeface in an interface is rare; generally, only one is needed. Where content consumption is the purpose, such as marketing materials, multiple typefaces may be warranted in order to differentiate levels of headings and body text.

Limit the number of font sizes to 4 in an interface, though more are occasionally needed. Consider the information hierarchy (e.g. titles versus descriptive text) and where you want the user to focus when choosing what information is displayed in the largest size. Be consistent with that size throughout the UI. Font sizing between panels can be different depending on pixel density versus the actual size of the screen but should strive to appear the same.

"Select" versus "Choose"

In general, the word "select" should be used when requesting that users make a selection (such as when picking a source, destination, or zone for control) over other words like "choose" or "pick." "Select" is less casual, and specific to situations where the options are of a set number.

Technical Terms and Jargon

Consider the audience, and make sure any terms used will be understood by them. Terms, such as "routing", which may be well understood by people in the audiovisual industry, are not well known outside of it. It's generally best to avoid using specialized terms when a more basic word would suffice. However, on an interface intended for technicians, specialized language provides more information and can be more precise.

An example would be expressing volume levels: on an interface intended for a normal user it may be best to provide a gauge showing the volume level at a position between a predetermined minimum (silent) and maximum, or as a percentage. On the technician interface, it may be more appropriate (and more specific) to express volume in decibels.

Numbers

It's better to use the actual numeral in an interface, and not the word, when referring to numbers: e.g. "3" instead of "three." In print materials, spell out single digit numbers, and use numerals for all others.

However, use common sense in situation where this would look awkward, e.g. “We have deployed this system in 10 locations, and actively maintain nine of them.”

When referring to larger, less specific amounts it may be appropriate to spell the number out regardless of the format: e.g. “PepperDash has worked with hundreds of clients.”

Confirmation Buttons

When a user takes an action and is prompted with a confirmation dialog, the buttons to confirm or cancel that action should be as clear as possible. Do not use “Yes/No” or “OK/Cancel” labels on buttons if another label can describe the action that will be triggered by pressing the button. For example, on a confirmation dialog used to end a meeting, the confirmation button should be labelled “End Meeting” instead of “OK.”

Color

Colors chosen should not only be pleasing to the eye but should also serve to clarify the functions of the elements in the UI.

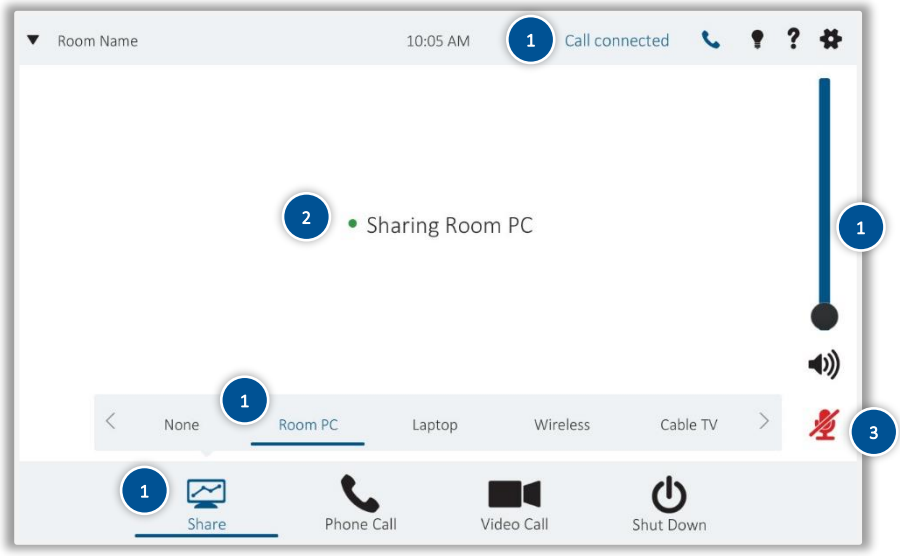

Figure 9
1) Blue indicates an active state; 2) Green indicates successful operation; 3) Red indicates critical feedback. All of these colors provide the required feedback, without overwhelming the simple palette of the interface.

Contrast

Contrasting elements draw the eye. Colors that are significantly lighter, darker, or brighter than the ground they are on draw attention because they strongly contrast with the nearby elements. This can, and should, be used to emphasize elements that enhance user interaction. Elements that are informational and passive should be relatively de-emphasized. An example is when a button is programmatically disabled and its transparency is reduced. This makes the colors on the button more closely match those on the ground, thus drawing less attention to it. While contrast is most often thought of in terms of values (brightness vs darkness) color, texture, and edges are all contrast vectors in design, and they all need to be considered when considering how contrast can draw the eye.

Consider the phone keypad in the following examples. In each example, the “Connect” button is different from the other buttons, and each time the eye is drawn to the button because of a different type of contrast. The most effective of these is then used to separate that button from the keypad in order to draw the eye to the most important button in that set.

Contrast Vectors

1) No contrast; 2) Edge; 3) Value; 4) Color; 5) Shape; 6) Size; 7) Location; 8) Color, Size and Location

1. There is no contrast between the connect button and the keypad buttons. As a result, the connect button appears non-distinct and could be difficult for users to locate.
2. The connect button has a white border around it, creating a contrasting edge. This makes the connect button stand out, but it is not clear why.
3. Value contrast is used to differentiate the connect button. Because this interface desaturates unavailable or disabled buttons, the contrasting values make the button appear to be disabled.
4. Contrasting color makes the button stand out. Because “Green” is associated with placing a call on contemporary interfaces, this not only calls attention to the button, but also helps to convey the meaning of the button.
5. The shape of the connect button contrasts with the keypad buttons because it has rounded corners. This difference is subtle, but still draws the eye. However, there is no lexical meaning to a contrast in shape in this interface, so the contrast does not serve much function.
6. Size is used to differentiate the connect button. This both differentiates the button and makes it easier to press due to a larger target area.
7. By moving the connect button from the keypad buttons, location is used as a contrast vector.
8. Finally, the most successful and intuitive contrast vectors are employed together. Color is used to convey meaning. Size is used for ease and to convey importance. And location distinguishes the unique functionality. While edge, shape, and value are all legitimate contrast vectors, they do not contribute to understanding or ease of use, so they are not employed.

When employing contrast to help draw the eye, rely on as few contrast vectors as are needed. Often, relying on value and/or color is sufficient. When choosing a contrasting color or value, be sure to check these with other colors and values that are used elsewhere.

Text

Choose the appropriate text color for the situation. Colored text should seldom be employed. Generally, white or light grey text should be used on a button with a dark value, and black or dark grey text should be used on a button with a lighter value to ensure legibility. Higher contrast can be employed to emphasize a button, while lower contrast can de-emphasize a button it should never sacrifice legibility.

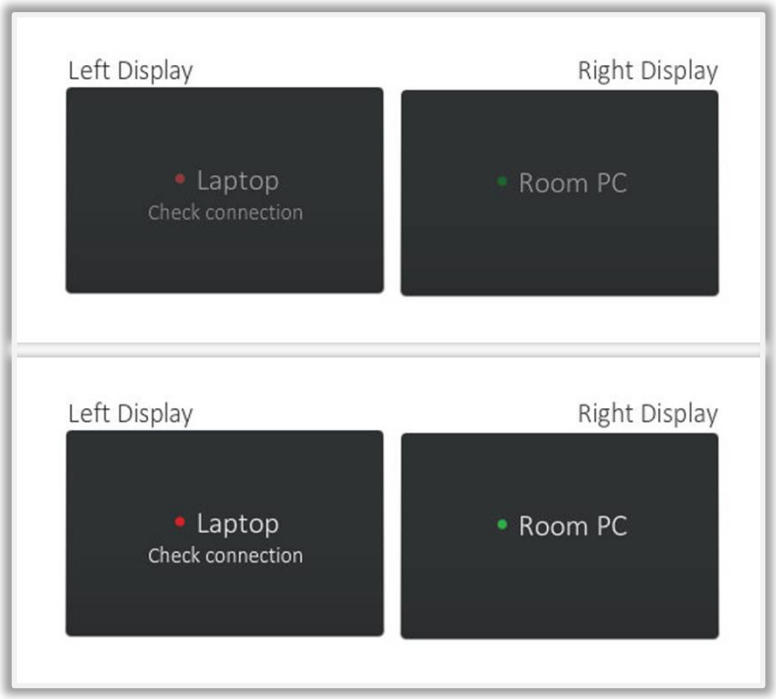

Figure 10
Top: Colors are too muted to offer sufficient contrast against dark background.
Bottom: Increased saturation and brighter whites make the colors legible.

Confirmation Buttons

Choosing the right color for confirmation buttons requires careful consideration. One line of thinking is that a green button should be used for the positive action (confirm, next, ok), while a red button should be used for the escape action (cancel, back, ignore). This is a great starting point but does not work in every scenario.

Consider that red is often used to warn the user of a potentially serious action that may be difficult, impossible, or time-consuming to undo (such as ending a call or powering off the room), while green can be used to express the less serious action. In the case of a dialog asking a user if they would like to shut down a room, these two modes of thinking are in conflict; "Shut Down" is both the positive choice (green button) and the more serious choice (red button). In these cases, using standard buttons (black or white) is a smarter design choice, as it emphasizes and enforces the text on the button and removes the pitfalls present in the two conflicting methods.

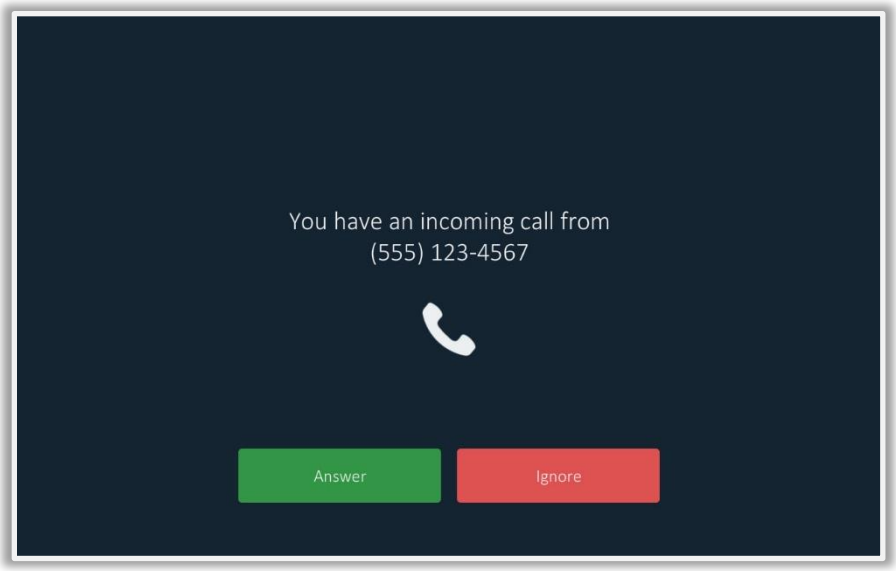

Figure 11
Answering the phone call is a positive action that carries no serious consequences (green). Ignoring the call is an escape action with potentially negative consequences (red).

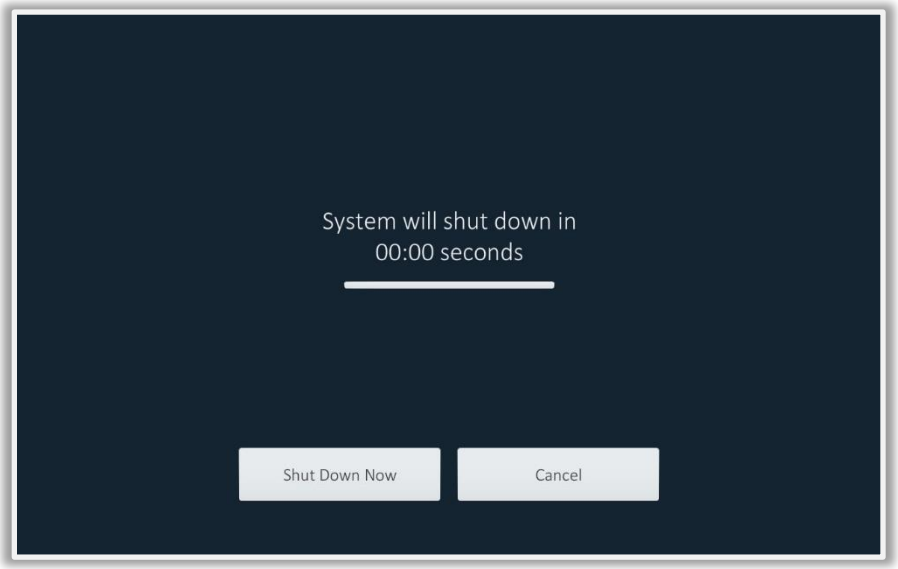

Figure 12
Shutting down the system is a positive action that does carry serious consequences, so both buttons are neutral colors.

Red

Use red sparingly in an interface to indicate an alert or warning. Some acceptable uses include a client logo, muted speaker and microphone status, emergency notifications, and escape buttons.

Limited Color Palette

Limit the number of colors in a UI design. If color is being used to indicate that an item is active or selected, then that color should be reserved for these functions, and there should be only one such color. Additional colors may be needed for background treatments, status indicators, or other iconography, but it's always best to use colors that do not compete with the primary color palette.

Iconography

Icons serve as a visual shorthand to quickly and universally communicate a potential activity, action, or command. Icons can be skeuomorphic in design, or simplified representations of real world objects. Regardless of their design, the purpose of an icon is always to convey meaning. Simplified iconography is more universal, and therefore more useful for a wider variety of applications.

Figure 13
Top: Skeuomorphic icons don't necessarily match real world counterparts, and can take extra time for users to interpret.
Bottom: Simplified iconography is more universal, and more easily accommodates a wide variety of active color states.

Geometric Perfection vs Visual Perfection

Icon weight and shape can change the way icons are perceived. Icons consisting of outlines are going to appear smaller than icons that have a solid fill. Icons with smaller surface area may appear shorter or narrower than icons that take up more area.

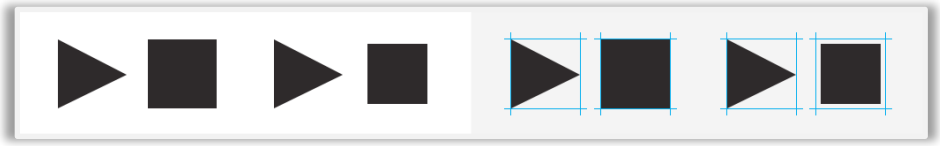

Figure 14
The pair of "stop" and "play" icons on the left are the same height and width in pixels resulting in a stop icon that appears larger than the play icon. Reducing the size of the stop icon helps correct this perception.

This same rule applies to the geometrical shapes that are used to create icons. Consider the Google "G" for example. In the image below, the icon on the left is the actual Google icon. The icon on the right is a modification using exact shapes. On the right is a detail view of the actual icon with cyan highlights to indicate how a slightly oblong circle is used to create the perception of a perfect circle inside the G. The orange highlight shows that the angle of the cap doesn't pass through the center of the circle and doesn't match up with the separation between the yellow and green areas. Despite this, it still feels more balanced than the geometrically "correct" version.

Figure 15
Left: The actual Google icon. Center: The same icon built using precise shapes. Right: A closeup analysis of the actual icon reveals how deviating from precise geometric shapes gives the overall perception of balance.

Consistency

Design icon sets with similar features (all rounded corners or all squared corners, all the same color, all sized the same etc.). If the icons exist on a button, then the buttons should be sized identically and appear similar as well. Occasionally, a particular button should be emphasized (such as making the record button red, or the play button larger in a set of transport controls), but care should be taken to ensure that they all feel part of a set.

Diversity of Design

Icons with different functions need to look distinctly different. Users should be able to differentiate between icons at-a-glance, even at reduced resolution. Labels may help with comprehension of these differences.

Figure 16

Left: Both icons communicate their meaning well at a large size. However, when these same icons are presented at a reduced resolution the differerence becomes harder to discern. The icons on the right however are visually distinct even at lower resolution.

Size and Complexity

Consider the final size of the icon when designing. Size and complexity share a positive relationship: the greater the size of an icon, the greater complexity may be afforded to it. All icons should appear that they were created for use at size.

Industry Standards

Use familiar icons when possible, such as commonly used symbols for power, tools, home, return, keyboard, volume, lights, delete and others. Don't introduce new icons for commonly used functions that people already have an associated icon they recognize.

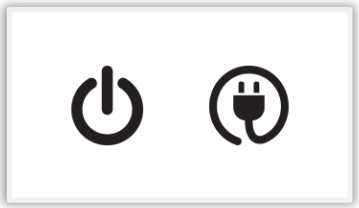

Figure 17
Left: The industry-standard power icon is instantly recognizable to most users. Right: Substituting a new power icon may cause confusion for users.

Buttons

Buttons are perhaps the most critical element of a User Interface, as they are how the user manipulates their AV experience.

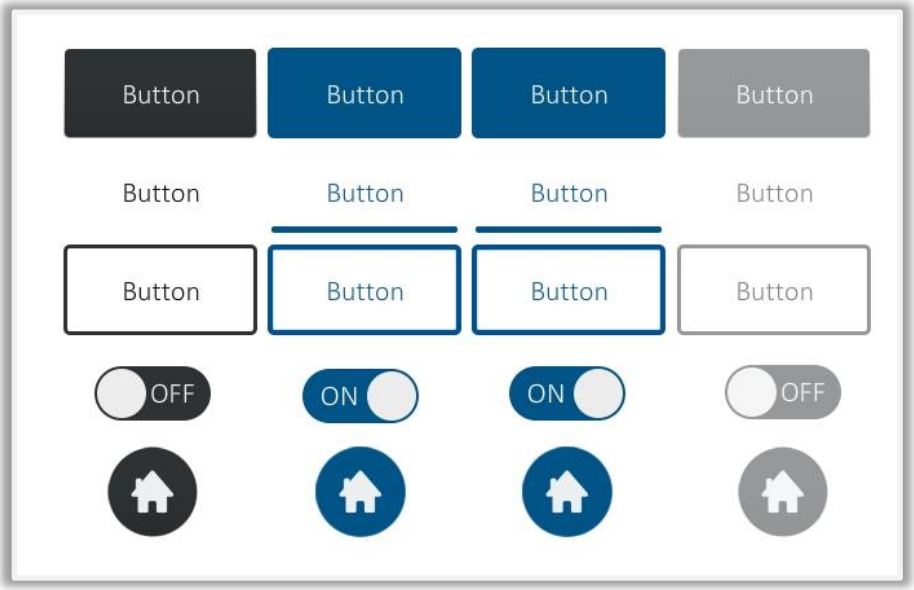

Figure 18
Left: Buttons in the inactive state; Middle-Left: Buttons in the pressed state; Middle-Right: Buttons in the active state (note that though the active and pressed states are identical, both must be defined.) Right: Buttons in the disabled state.

Icon and Text Legibility

Consider the icons and text that may appear on the button. Make sure it serves as a suitable platform on which to frame both. Don't design overly busy or gradated buttons that will make it difficult to see text and icons. Don't create button states that can't support the same material.

Active State Color

All buttons should have the same color active state, with the exception of special circumstance buttons (Call/Hang up, Record, etc.). When possible, volume gauge fills should use this same active color.

Button Styles

All buttons belonging to a particular category should share the same style. For example, all source buttons should belong to a defined button style, and all device controls should belong to a particular button style. All button styles used should complement each other. Lastly, it's best to use the smallest amount of varying button styles throughout a UI to create a cohesive look. Not all button types need a distinct style.

Interactivity

Buttons should be inviting and obvious that they are meant to be pressed. Use caution when using buttons with text only on a transparent button, as it may not be obvious it's interactive. Pairing an icon with text on a transparent button often communicates that it is interactive.

Dimensionality

Effectively using Z-space can create a pleasing sense of depth, define different areas of the UI, and provide visual clues to the user.

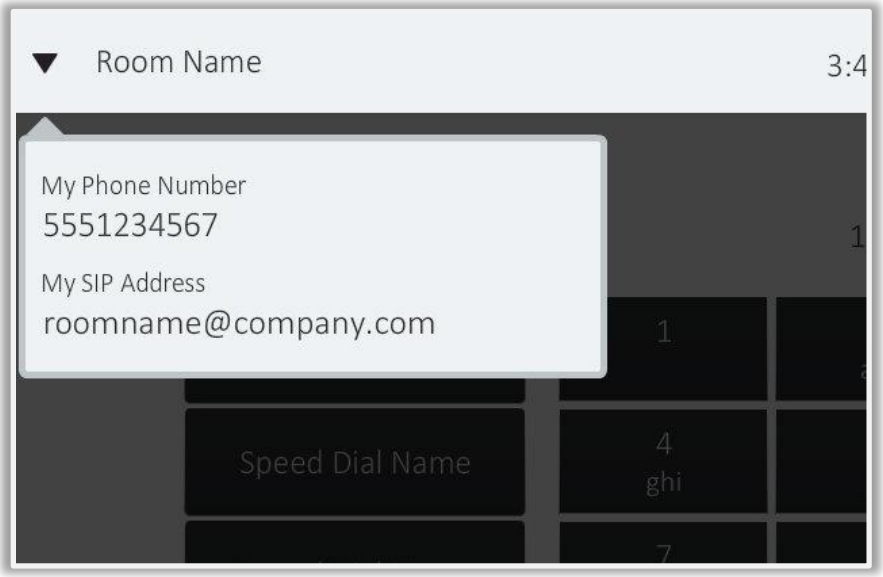

Figure 19
A modal dialog that appears on top of other controls blocks out the disabled controls with a shadowed background. This defines the Z-ordering for users, while simultaneously blocking unintended interactions form occurring.

Global Light

Ensure that any shadows cast by a perceived single source remain consistent across all elements. Don't create shadows that give the appearance of conflicting light sources.

Effectively using Z-space can create a pleasing sense of depth, define different areas of the UI, and provide visual clues to the user.

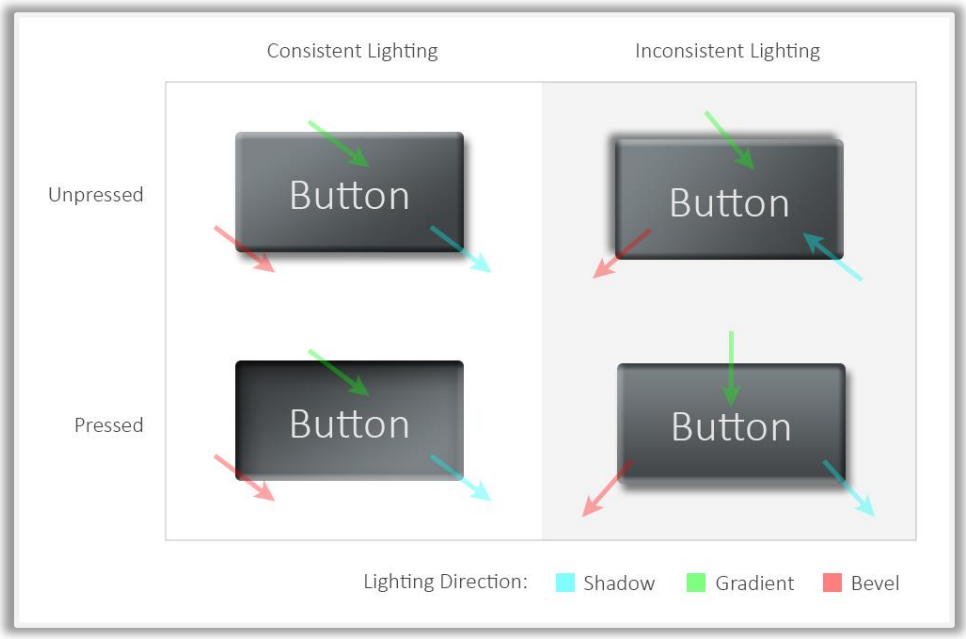

Figure 20
Consistent lighting is used to define states of a visual element, while inconsistent lighting creates confusion.

Button States

Be consistent when using shadows on buttons states. The diffusion and darkness of a shadow cast by a button can create the appearance of height from the background. Variations in this distance can simulate a button press. If this technique is used, ensure that it is used consistently (Example: a shorter shadow distance should be the pressed state across all buttons). Be consistent when using shadows to define a button state.

Effectively using Z-space can create a pleasing sense of depth, define different areas of the UI, and provide visual clues to the user.

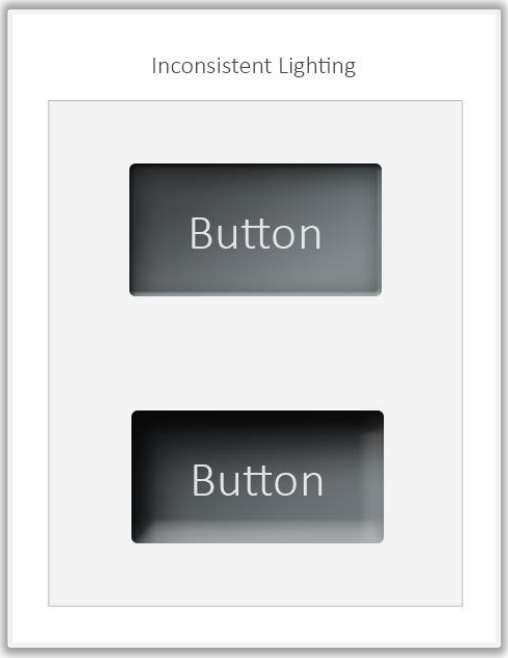

Figure 21
Treatments on the bottom button imply that the button is pressed more deeply. The different treatments create ambiguity regarding the state of the two buttons.

Unobtrusive

Ensure that the dimensionality of the panel does not take center stage. Any shadows should be tight and unobtrusive. They should be used for emphasis and not draw attention to themselves. Don't allow a shadow to take up too much space, draw too much attention, or infringe on other elements.

Popups & Modals

Popup pages with notifications or environmental controls (for example) need to appear as if they are in the foreground. Ensure modal dialogs protect users from interacting with interfering controls. Often, this is achieved by making these dialogs the size of the entire page with either a solid background or a semi-transparent background, to indicate that users must close out of the popup to return to the controls on that page. Don't have a popup page with a fully transparent background, causing confusion between popup controls and controls on the main page.

Layout

When positioning elements on the page, not only must interactive elements themselves be proportional and symmetrical, but the white space around those elements must also be carefully proportioned.

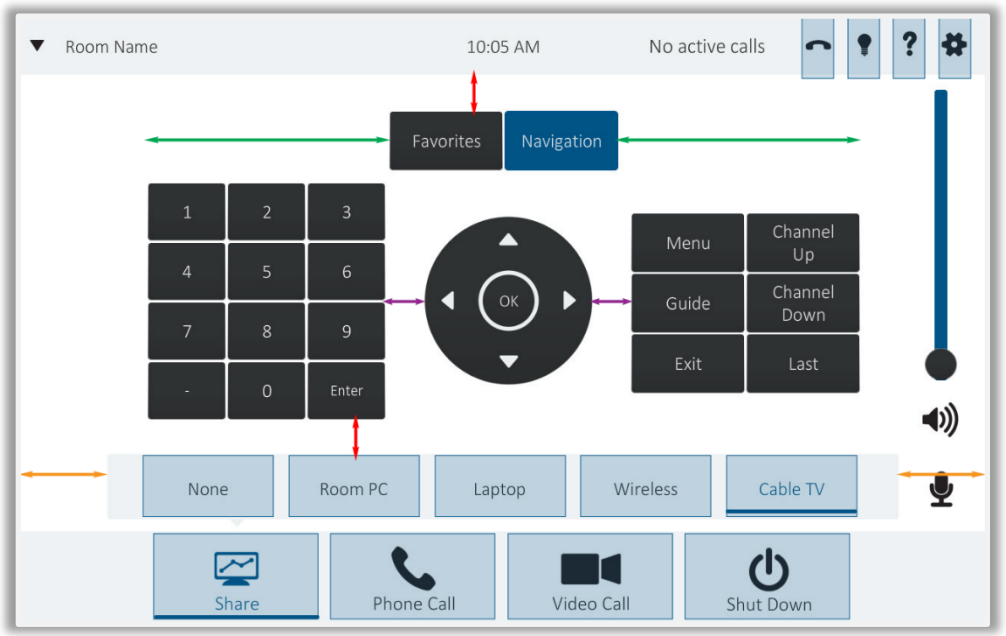

Figure 22
Arrows indicate equal amounts of white space to balance objects on the interface. Boxes indicate equally sized boundaries for interactive elements.

White Space

Correctly spacing elements on a page helps users categorize those elements into families, without the need for the additional visual clutter that comes with boundary boxes and container elements. It is important to measure the dimensions of the white space in an interface when designing a layout.

Figure 23
Top: Lack of white space between buttons makes it difficult to group buttons into functional groupings. Middle: Horizontal white space obfuscates the grouping of buttons; Bottom: Vertical white space allows the user to correctly group the buttons.

Text Alignment

Keep alignment consistent across all similar elements. If a button is left-aligned, all other buttons of that type should be left-aligned. Don't use different alignments for similar elements.

Edges

Position text and interactive elements such as buttons away from the edge of the panel.

Direction of Actions

Lay out areas of the UI in an intuitive manner when there are tasks that require multiple button presses (such as selecting a source and then a destination when routing). Proprioceptive usability must be considered when deciding on a navigation paradigm. When a user presses buttons toward the top of an interface, the user's arm obscures more of the interface than when user's interact with the bottom. This is more noticeable on wall or table-mounted devices than with handheld devices due to the distance between the viewer and the control surface.

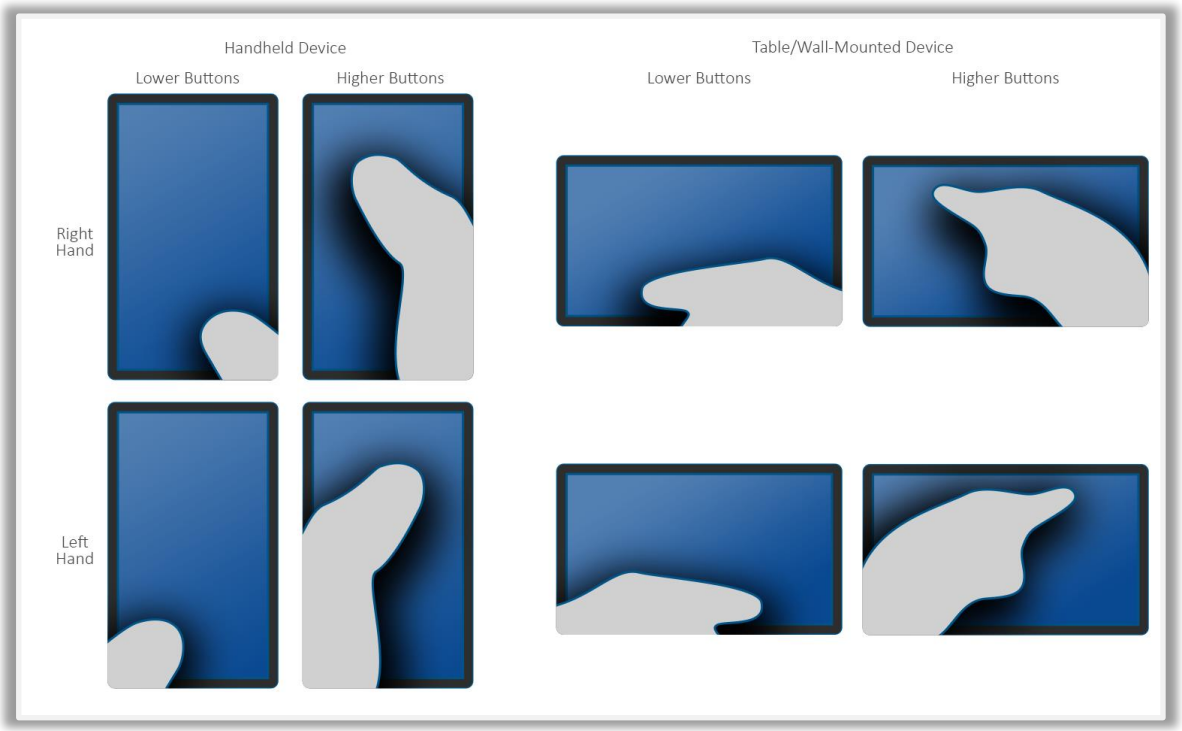

Figure 24
Grey shapes above represent the user's body, and black shadows indicate areas that are obscured by the user's interaction with a device. More of an interface is obscured when users interact with the top of a device.

Size and Position Consistency

Keep size and position consistent for elements that exist on more than one page, such as navigation and settings buttons and bars. This will keep these elements from "jumping" when a user switches between pages.

Navigation and Settings

Create a clear delineation between global functions such as navigation and settings, and those that are specific to the current page. The user should be able to anticipate which areas of the page will change and which areas will not before they perform an action.

Volume

The industry standard location for these controls is on the bottom of the page or on the right side.

Notice of Ownership and Copyright

Notice of Confidentiality

Notice of Trademark and Servicemark

A higher level of control™