8

Look and Feel

When the Apple Macintosh was first released the terms “look” and “feel” entered computing culture. Though Apple did not invent these terms they made them real for the computing public. The Macintosh enshrined the concept of a common look and feel across all applications as they had been done in the Xerox Star. The purpose of a common look and feel across all applications is to simplify usability of an interface[1].

In particular a consistent look and feel reduces the learning of a new interface. This consistency aids learning in two ways. If an application is internally consistent (has a consistent look and feel across the application) then learning one feature also teaches the user how to access many other features. If different features have different interactive mechanisms then each must be learned independently. External consistency (features work the same across all or many applications) also aids learning because users can reuse knowledge that they gained on other applications and need not learn yet another approach.

External and internal consistency also aid in remembering how to use an application’s interface. Most computer users have 2-3 applications that they use regularly and many more that they use occasionally. Even within a user’s primary applications there are features that are used infrequently. A consistent look and feel aids a user in rapidly remembering and reusing these features.

A consistent look and feel is also important to the effectiveness of an interface. This chapter will discuss some physiological reasons why consistency increases speed and effectiveness of use. We will first address what it means to be consistent and what kinds of consistency are important. We will then discuss the look of an interface, which is implemented in the view. Lastly we will discuss the interactive feel, which is implemented in the controller.

Consistency

Before launching into the issues of look and feel we need to attend to the meaning of consistency. Grudin has argued clearly that trivial notions of consistency can actually damage the usability of an application[2]. He provides a compelling example of where one should store the knives in a home. Consistency would dictate that all knives should be stored together so that whenever you want a knife you know right where to find it. However, carving knives are frequently stored in a block stand within easy reach and where their edges are protected. The good silver knives are stored in a special box in the same place as the china and crystal. Putty knives are stored in the garage with the other tools. His primary point is that consistency should be consistency of use rather than consistency of thing.

To translate these ideas into user interfaces, consider the process of selecting a file to open. Applications should use the same mechanisms for finding files in all parts of the program. Using one technique in one place and another in a different place would be inconsistent. However, we must consider the use of those files. If a user is selecting images (which are stored in files), we can provide a file open dialog to select them. However, the standard file open dialog uses file names, sizes and types to select files. Users frequently select images by what they look like. Though we may want to allow images to be selected using the file dialog, this new use may imply that selection from small thumbnail images would be more appropriate. If we decide to provide a special image selection technique then that technique should be available wherever images are selected and not just in some places.

Look

Designing the “look” of an interface primarily involves the view. Designing the look is best done visually before any code is written. Java and C# may be helpful in designing a model but they are terrible tools for developing a look. Designing a look is not completely independent of the “feel” implemented by the controller. How an object is displayed has a lot to do with how we interact with that object. Displaying line segments in a drawing where they can be selected and their endpoints can be dragged around is very different from displaying line segments in a table showing each of the four coordinate values of the two endpoints. The first view shows you how the resulting drawing will appear and the second gives you exact control of coordinates. Which of the two is appropriate depends very much on the task at hand.

Before discussing principles of visual design we first will review the physiology of the eye and vision system. We show how these concepts influence visual consistency of a look. We then discuss the concept of affordances and how they influence the look of an application. We then discuss visual design concepts and how they can be used to draw the user's attention to what is most important. Lastly we discuss presentation of the state of the interface.

Physiology of the Eye

It is helpful when designing a look to understand some basic principles of how the eye functions. Our primary goal with a design is to overcome the gulf of evaluation by bringing as much useful information to the user through their eyes as rapidly as possible. Sight sensing begins with the retina where we find two types of sensors: rods and codes. Rods are sensitive to a broad spectrum of light and therefore have no ability to discriminate among colors. Rods are particularly sensitive to dim light and provide high amplification. Cones are of three types, each sensitive to red, green or blue. Based on the particular type of cone that is stimulated, we sense a different primary color. The absence of appropriate pigment in a particular type of code will produce color blindness where someone cannot distinguish one of the primary colors. The ratio of rods to cones[3] is about 20 to 1. The numbers of various types of cones are not uniform. This distribution of cones is red (64%), green (32%) and blue (2%). Though there are very few blue cones, they are more sensitive and have higher amplification. Their lack of numbers does, however, reduce the ability to sense fine detail.

The rods and codes are not uniformly distributed across the retina. There is a region in the center of the retina where most of the cones are concentrated. This is called the macula. Within the macula there is an even more concentrated region called the fovea. This small portion of the retina is where "reading vision" occurs. To understand this effect, focus your eyes on one word in the middle of a paragraph on this page. Without moving your eyes attempt to read words in the neighborhood of your focus word. We feel that we see the world in high resolution because we can rapidly move our eyes so that the fovea is focused on the area of interest. In reality we have relatively low resolution perception of most of what we see at a particular instant. This is one of the reasons photography is never as "real" as being there. If we are present in person we can focus on whatever we desire and see it in high resolution. With many photographs they are either highly focused on a particular thing providing high resolution but no ability to look around or they are wide with low resolution providing little ability to study what we want.

Another property of the distribution of rods and cones is that the rods are generally scattered around the periphery (outside the macula) while the cones are concentrated in the fovea. This means that our peripheral vision is mostly gray scale rather than color. Even though the ratio of rods to cones is 20 to 1 the ratio of the peripheral area to macular area is much higher than that. This produces the low resolution of peripheral vision.

We exploit the differences between the periphery and the macula by rapid eye movement. This allows us to rapidly focus the fovea wherever we want. However, this movement is not continuous. The eye moves in saccades or periods of move and stop, move and stop. Each saccade takes about 200 milliseconds or 5 times per second. The periphery and the fovea work together in controlling eye movement. Because of the periphery the eye muscle control system knows where to move to focus on some item detected in the periphery. The periphery also helps in maintaining context and alignment of images as the fovea jumps around.

The structure of the visual process in the brain is also important. A major component of visual understanding is edge detection. One of the first stages in the visual cortex of the brain is to detect the location, orientation and strength of edges. The dependence upon edges in our understanding of an image is why line drawings or cartoon sketches are just as recognizable to us as the original image. Texture recognition is closely related to edge detection. These are primary visual mechanisms for segmenting an image into meaningful parts. A related process is the periphery's ability to detect motion. To protect ourselves from danger there are reflex processes that translate detection of unexpected motion into a flinch or jump with very little cognitive effort. This keeps us safe from falling or flying objects but makes it very difficult to ignore objects that are flashing or jumping.

Visual Consistency

One of the most time consuming parts of using an interface is finding the needed controls or visual objects. Detailed study of labels or icons requires that fovea be focused on them and the fovea can only be moved 5 times per second. The word processor used to write this chapter has approximately 100 icons or widgets available on the screen. This does not count pull-down menu items. To scan each of them with my fovea as fast as possible would require at least 20 seconds. To achieve rapid visual use of an interface we must have faster means for locating items than a foveal scan.

A most important mechanism for rapidly scanning a display is to place items in a consistent location. Using the visual periphery as an alignment mechanism, the eye can rapidly switch to a known location in 1 to 3 saccades. Very common items, such as new document, open, save and print, are always located in the same position and one can find them without scanning. Using the same icons for the same concepts also improves scan time. The low resolution periphery can rapidly locate regions that are consistent with the desired shape and allow saccades to only jump to those visual locations rather than saccade to each icon trying to understand it. The consistent look of a tool-bar, as shown in figure 8.1, also helps the visual scan. The blended look has a nice appeal and the strong dark line that it forms allows the low resolution periphery of the retina to guide a rapid scan of the tool-bar.

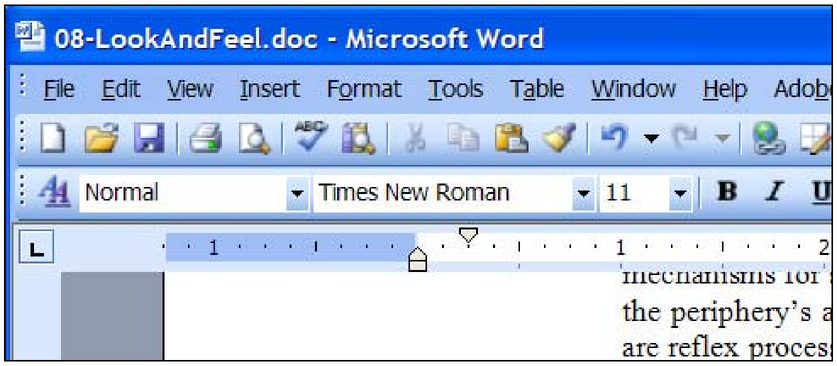

08-LookAndFeel.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help Adob
Normal
Times New Roman
11
B I U
mechanisms for
the periphery's a
are reflex proces

Figure 8.1 – Consistent icons and tool-bars

Figure 8.2 shows the same interface fragment only blurred and gray scale as the periphery would see it. Note that one can readily find the window title, menu bar and icons. The consistent look of the drop-down combo boxes for style and font are easily identified in the blurred peripheral image. The icons for Bold and Italic are also distinctively recognized. However, without a consistent interface look such recognition in the periphery would be impossible.

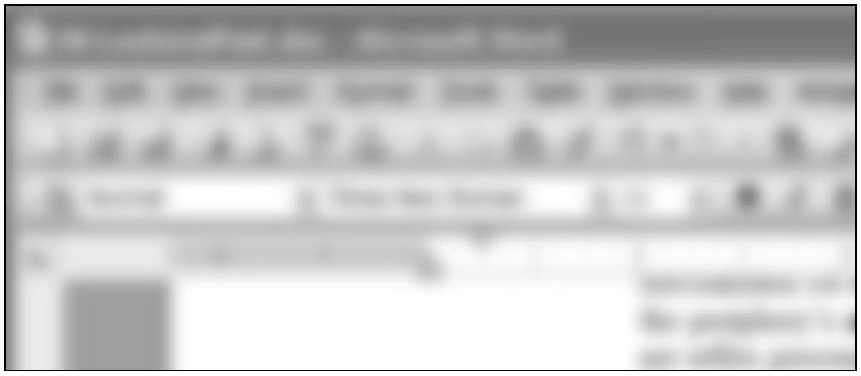


Figure 8.2 – Peripheral view of an interface

Affordances

One of the key ideas popularized by Don Norman[4] is the concept of affordances. An affordance is something the leads you to know how to interact with an object. Nobody needs training on which end of a hammer to hold. The wrong end simply does not lend itself to holding. A good example of an affordance is the door bar shown in figure 8.3. This bar only presses on one side of the door, which is the side that can be opened. Many commercial door mechanisms have a bar that crosses the entire width of the door with a uniform look all the way across. The user is left to guess which side of the door will actually open. Pressing on the hinge side does nothing. The partial bar has a good affordance (doing the obvious thing makes it work) and the full bar has a poor affordance (guess which side to press).

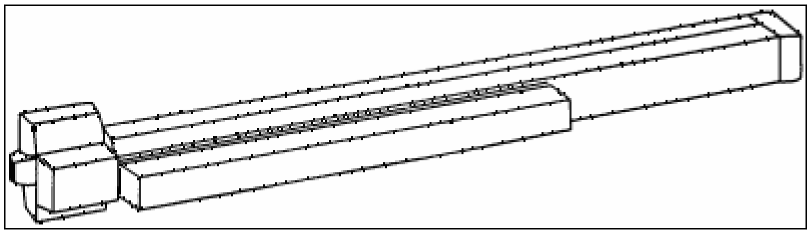


Figure 8.3 – Affordance of a door opening bar

Many of the early visual designs of user interfaces drew their affordances from physical objects. Buttons have a three dimensional look that makes them visually protrude from the surface of the interface while labels are given a flat look that makes them appear painted onto the surface. Scroll-bars have little traction marks like physical sliders to provide the look of something that can be moved with a finger.

Realistic ties to the physical world may not be effective. User interfaces that looked like rooms with documents lying on tables and pictures of real file cabinets were never widely accepted. There are three reasons these highly realistic affordances did not succeed. The first is that the physical behavior did not translate well to screen behavior. For example there is no “pull” action on a screen that maps well to pulling open a drawer. The user may see a drawer and yet suffer a gulf of execution problem by not knowing what will open a drawer on the screen. The second problem is the visual scan. It is frequently much easier for the visual periphery to scan for a simple stylized shape than a more complex realistic shape. Lastly the realistic views tend to take more screen real estate which is generally scarce.

The most common form of affordance today is consistency with other interfaces. If a widget looks like the other scroll-bars or buttons that the user has seen, then users are likely to know how to use it. If it looks like other widgets yet behaves very differently, this can be a negative affordance. The goal is to design a look that most users will understand how to use because it looks like other things that they have seen and behaves in a similar fashion. One very useful affordance is if items react visually when the mouse passes over them. This indicates to the user that such items are active and will respond to input.

Visual Design of Attention

Scanning a large screen to locate necessary items can be time consuming and the look should be designed to minimize that time. To do this we can use concepts from visual design that help draw visual attention to those things that are most important.

The most important visual attention concept is the reading scan order. Being taught to read defines a natural order for perusing a page. Among European languages the scan pattern is left to right, top to bottom. Thus the first place that a reader of a European language will look is at the upper left corner and then across the top. For European language users that is where the most important information and controls should be placed to be found most quickly. The last place that such users will look is the bottom right corner. However, Hebrew and Arabic speakers read right to left, top to bottom. The scan order is thus reflected horizontally. Among some Asian languages the scan of a line of text is vertical rather than horizontal. Schools spend a lot of time teaching people to read and our visual design should use that training rather than fight against it.

An obvious rule of attention is that larger items attract more attention than smaller items. That is why section headings and titles are generally larger and/or thicker than normal text. It makes it easy to scan through the main points. Figure 8.4 shows a blurred region of text as the retina’s periphery would see it. Note that the section heading is readily identified. This means that the eye can find the heading in a single saccade using the peripheral information to target the correct location. Care should be taken in overusing size. If the size of the text becomes larger than the foveal region then it becomes harder to read rather than easier. The user must move away from the text to read it comfortably.

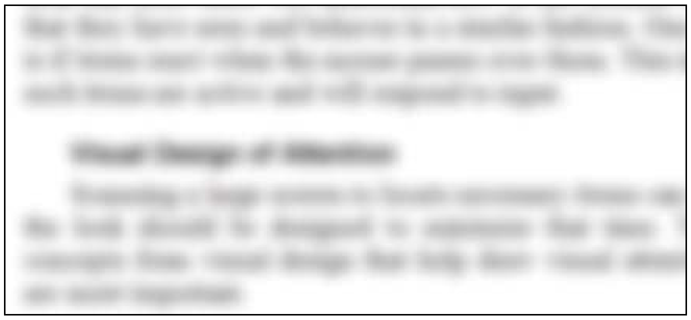
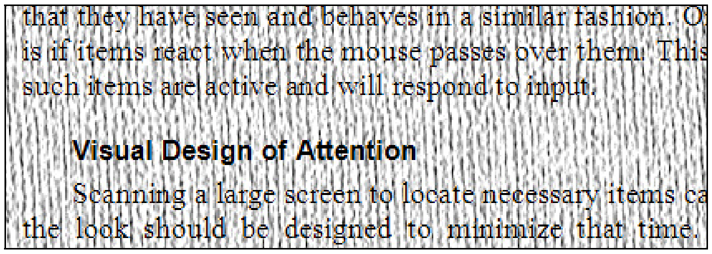


Figure 8.4 – Periphery view of a heading

A third attention principle involves texture. The edge detecting visual process is drawn to textured areas as more interesting than smooth toned areas. Surrounding active objects with smooth "white space" will set them off and make them easier to find. Adding lots of borders and decorations will draw attention away from the important information. White space leads the eye towards the interesting information. This principle is generally violated in two ways. The first is to put borders around items rather than using empty space to set them off. The result is a busy look that makes it difficult to find important information. Only use borders when you really need them and make them lower contrast than the key information. The second mistake is the excessive use of textured backgrounds. This is a particular problem in web-page design. The background texture interferes with the important information. Figure 8.5 shows a text segment with a heavy texture behind it. Not only does the texture draw attention away from the text, it also makes it much harder to read because the texture edges distract from the character edges.



that they have seen and behaves in a similar fashion. O
is if items react when the mouse passes over them. This
such items are active and will respond to input.

Visual Design of Attention

Scanning a large screen to locate necessary items ca
the look should be designed to minimize that time.

Figure 8.5 – Textured background

The next attention principle is contrast and light. The eye is drawn to high contrasts and to light regions in preference to dark regions. The contrast comes from the edge detection process of the visual system. High contrast produces strong edges from which shapes are recognized. The preference for light comes from the retina’s function as a light detector. The use of contrast also guides the use of color. Color is best used as a highlight of colored vs. not colored or high vs. low saturation. Color is less effective to encode information although it can be used as such. A colored area in an otherwise desaturated gray area will draw some attention. However, remember that the periphery is basically gray scale in its sensing. Color saturation should never be used alone to highlight a region. There must also be a difference in brightness. Brightness contrast accommodates the retina’s periphery as well as color-blind users. Figure 8.6 shows a text message printed blue on red. This has problems in several ways. The first, shown on the right is that the periphery with little color sensing would have trouble even finding the text because of poor contrast. The second is that the text is in blue which has the fewest of all sensors in the eye. Because there is no contrast in brightness the rods cannot help at all and because there is high saturation the green cones are useless also. Contrast should first be ensured on the brightness axis so that the rods can help. Secondly contrast should be ensured on the saturation scale so that all of the cones are involved. Contrast in hue alone has poor visual value.

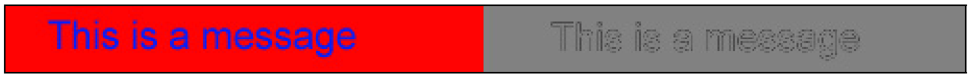


Figure 8.6 – Poor color contrast

Presentation of state

The look must clearly present the state of the widget. In most cases the visual look will present enabled/disabled, active/inactive and an echo of the current state of the widget.

A widget is enabled if it is currently appropriate for that widget to accept input. Figure 8.7 shows both enabled and disabled menu items. We could simply remove all disabled items from the menu, but this causes other problems. If a user is searching for an item and it is missing then they may search the entire user interface, including the menu structure, looking for something they know that they have seen before. This is very frustrating. If, however, they see the item but see that it is disabled then they at least know what is going on. A very frustrating feature of most user interfaces is that disabled items do not show tool-tips that explain why the item is disabled and what can be done about it. This creates a serious gulf of execution problem because the user is confused about what will enable the desired item.

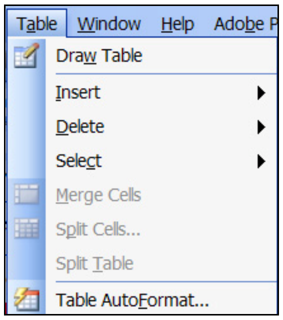


Figure 8.7 – Enabled and disabled menu items

A second problem is visual consistency. Removing an item from the interface will generally cause other items to move around in the interface. Users get very good at remembering where they last saw an item. If the removal of another item causes a change of position, this can be very confusing.

For these reasons we disable widgets that are temporarily inappropriate rather than removing them. The disabled look should generally display with less attention attraction than the enabled look. It is very common to show a disabled state using lower contrast than the enabled look. In figure 8.7 the disabled items are labeled in gray (lower contrast) rather than in black. This makes it easy to visually skip over them while still being able to identify what they are.

A widget should also indicate whether it is active or inactive. A widget is active generally when it currently has the mouse or the key focus. When the mouse moves over a widget and the widget's look changes from inactive to active, the user is then aware that this is something that can receive input. Figure 8.8 shows a tool bar where the icon under the mouse is shown with a highlighted background and a tool-tip (active state) while the other icons have no such highlight (inactive). Figure 8.9 shows a list of style names with the one under the mouse displayed with a border and an indicator for a pull-down menu. The other items are shown only as names.

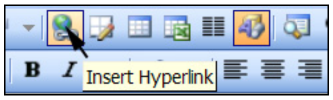


Figure 8.8 – Active/inactive tool bar icons

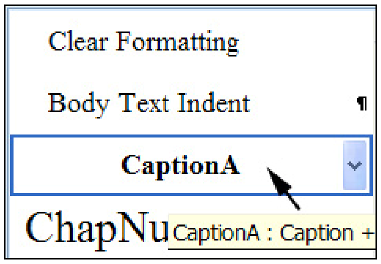


Figure 8.9 – Active/inactive style choices

The echo of a widget displays the state of the widget in a form that is readily perceived by the user. An appropriate echo will depend upon the purpose of the widget. Figure 8.10 shows two versions of a vertical scroll bar, one showing the current value and one without. When using a scroll-bar to pan around an image, the actual number of the current value is generally not helpful. The relative position in the image is the most important information and thus we would use the version on the left. However, if scrolling through a program listing where line numbers are important, the scroll-bar echo on the right may be more appropriate. We should carefully design a look that will clearly echo the information that is most necessary for the task at hand.

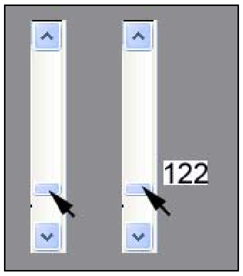


Figure 8.10 – Two scroll-bar echoes

Feel

The feel of an interface defines how it interactively behaves. In particular it specifies the design of the controller. A consistent feel is even more important than a consistent look. Figure 8.11 shows three different styles for a scroll bar. The left and middle variants show a similar appearance to most scroll bars although they are drawn differently from the look of most toolkits. The one on the right has a somewhat different look in that the step arrows are attached to the slider rather than fixed at the ends. However, most users when seeing one of these will recognize it as a scroll bar and will know how to try to use it. Clicking on the arrows will step the bar. Dragging the slider will move to a new position clicking in the empty space will page the scroll bar up or down.

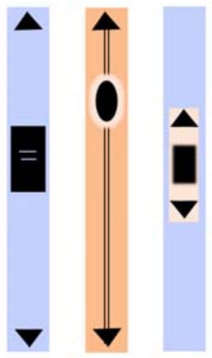


Figure 8.11 – Scroll variations

A first time user would try these techniques because they are consistent with the ways that scroll-bars behave on all major computer systems. However, there is nothing about the look of these scroll bars that says that they must work in this way. We could design our controller so that the slider does not drag at all but is moved by clicking the left mouse button for page down and the right mouse button for page up. We could redefine clicking in the empty space as moving the slider directly to that position rather than paging up or down. We could define right click in the empty space to go directly to the clicked position with left click performing a page operation. We could also define right click to open a small text box where we type the value rather than clicking and dragging to get there.

All of these controller behaviors are consistent with the three looks in figure 8.11 but they all would be extremely confusing to the average computer user. Before the Macintosh became popular with its uniform look and feel each individual program created its own scroll bar behavior. This created a huge gulf of execution because the feel of an interface is invisible leaving the user to guess at the appropriate inputs for manipulating a widget. Users can sort out the differences in look in figure 8.11 with a little thought. Users have no recourse in discovering a new feel. Because of this it is extremely important that the feel be consistent where possible across applications.

A consistent feel is also important to the *perceived safety* of the interface. One facet of an easily learned interface design is that the user can discover most of the features by exploring the interface rather than by reading the manual or help pages. To foster such exploration the user must feel safe in the interface. This perception of safety is fostered by predictability of behavior. If the user performs some action they should have confidence in what might possibly happen. This confidence and predictability not only simplifies learning but also increase speed of use because the user is not burdened with guarding against mistakes. If the feel is not consistent within an application and across applications, the user must slow down and be more careful.

When the Macintosh was introduced it popularized the pull-down menu. The feel for such menus was to perform mouse-down on the menu header at which point the menu would appear. The user would then drag the mouse (holding down the button) until the desired menu item was highlighted and then release the button to select the item. Macintosh users became very comfortable with this feel. When Windows was first introduced the menu bar looked similar to the Macintosh but the feel consisted of clicking on the menu header, moving the mouse with the button up and then clicking on the desired item. When Macintosh users attempted to use Windows systems there was chaos. Macintosh users would push down the mouse button causing the menu to appear. However, when dragging the mouse down to select an item the menu would disappear as soon as the mouse left the menu header button. This “peek-a-boo” menu behavior was very disconcerting to Mac users moving to Windows. Microsoft recognized the problem and released a new menu implementation that worked with both the original Windows feel and the dragging feel of the Macintosh. They very much wanted ex-Mac users to feel comfortable on their system.

There are occasions when an application has some new concept that does not generally exist in other applications and a new interactive technique must be designed. In those situations the feel should be consistent within the application and the developers must shoulder the burden of training users in the new technique.

For graphical user interfaces there are a small handful of mouse behaviors that make up the majority of interactive inputs. This alphabet of behaviors is:

- <u>Click</u> – mouse button down and then up with minimal mouse movement. This is always used to cause an action to occur or to select an object.
- <u>DoubleClick</u> – two clicks in rapid succession with minimum mouse movement. Most systems allow users to set the time and movement constants to adjust for the dexterity limitations of various users. Applications should use these system-wide control settings rather

than circumvent them. DoubleClick is always used to “open” the object being referenced to show more information about it.

- RightClick – a click with the alternate (usually the right) mouse button. This will bring up a menu of commands that can be applied to the object being clicked.
- Drag – a mouse-down followed by extended mouse movement followed by mouse-up. This has a variety of uses: moving an object, selecting a region or set of objects, or creating an object. Usually the goal is to provide a start point and an end point for some action. The normal technique is to echo what the action would be if the mouse up occurred at the current mouse move position.

The fact that there are so few mouse behaviors led the designers of model-view-controller to separate the controller into its own class so that the code could be reused. However, the simplicity of these behaviors and the complexity of integrating them smoothly with the view, has led to the integration of the controller with the view into a single class.

Summary

The look and feel of an interface define its appeal, usability and learnability. The perceptual structure of the visual system dictates several principles to guide the look. Consistency of both the look and the feel is important for the learnability and usability of most applications. Being consistent with other applications in the same environment can greatly simplify the user experience. Consistent feel is extremely important because the feel is invisible to the user.

---

[1] Polson, P. “The Consequences of Consistent and Inconsistent User Interfaces” *Cognitive Science and its Applications for Human-Computer Interaction*, Lawrence Erlbaum, Hillsdale, N. J. (1988).

[2] Grudin, J., “The Case Against User Interface Consistency”. *Commun. ACM* 32, 10 (Oct. 1989), 1164-1173.

[3] Kandel, E. R., Schwartz, J. H., and Jessell, T. M., *Principles of Neural Science*, McGraw Hill (2000).

[4] Norman, D. A., *The Design of Everyday Things*, Basic Books (2002).