Drag and Drop
Human Interface Guidelines

September 14, 1993

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
408-996-1010

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

CHAPTER 1

Drag and Drop Human Interface Guidelines

Contents

Design Overview

Introduction

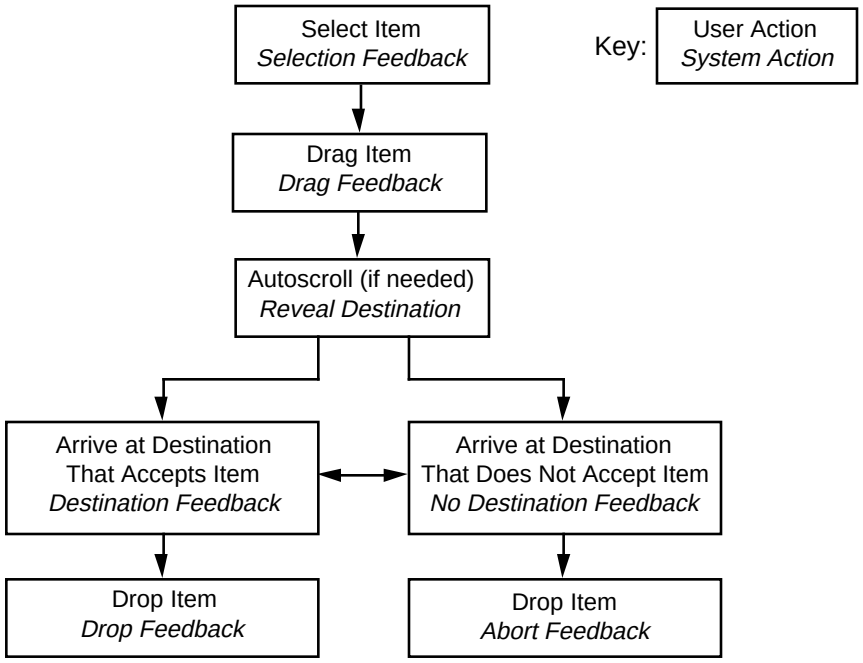
One of the Macintosh's key design characteristics is the concept of **direct manipulation**. System 7 extends this concept by allowing users to directly manipulate a document icon and "drop" it onto an application icon, which results in the document being opened with the application. This general technique of dragging an item and dropping it on a suitable destination is called **drag and drop**.

The Drag Manager from Apple Computer, Inc. extends the direct manipulation capabilities of the Macintosh human interface even further by providing a engineering framework for drag-and-drop sequences among windows and applications. This document specifies the **look and feel** of the behaviors supported by the Drag Manager, and provides human interface guidelines for developers who want to support drag and drop in their software. For convenience, these guidelines take a mouse-centric view of drag and drop; you should not overlook alternate input devices, such as pens and trackballs, as equivalent to using the mouse.

Interaction Model

A drag-and-drop sequence has several components, as indicated by the interaction model shown in Figure 1-1:

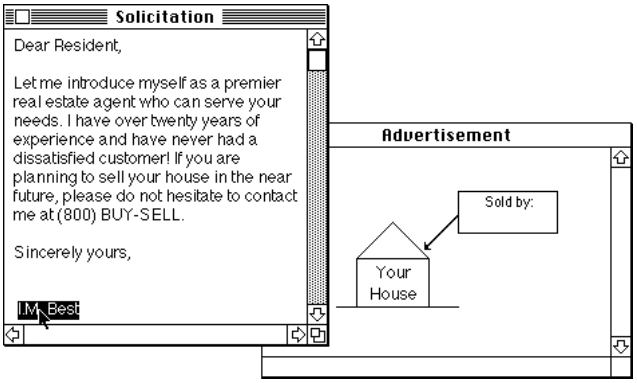
Figure 1-1 Drag-and-Drop Interaction Model



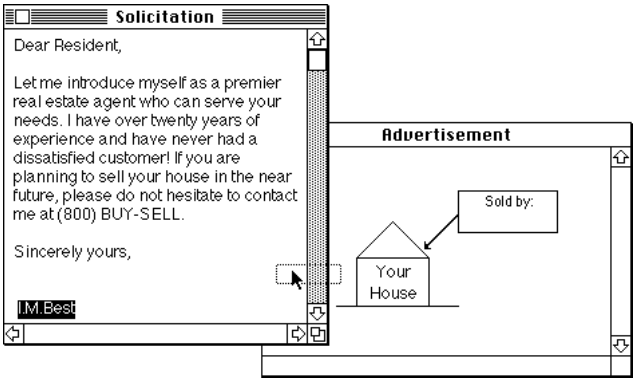
While this interaction model is not new, the Drag Manager enables a much richer set of dragged items and useful destinations. Up until now, drag-and-drop sequences that span windows have been limited to items characterized as containers. For example, documents are containers of content, and these documents could be dragged only across folder windows, disk windows, and the desktop in the Finder. Also, dragging content itself (such as graphics) has been limited to a single window. For example, MacDraw objects can be dragged to another location in the same window, but cannot be dragged to another MacDraw window. The Drag Manager eliminates these limitations; ideally, users should be able to drag any content from any window to any other window that accepts the content's type. This new capability leads to a generalization of the interaction model shown in Figure 1-1 on the previous page, where "items" and "destinations" take on broader meanings. For example, the scenario shown in Figure 1-2 below is now possible:

Figure 1-2 Dragging Text from One Application to Another

Step 1: Select text before dragging



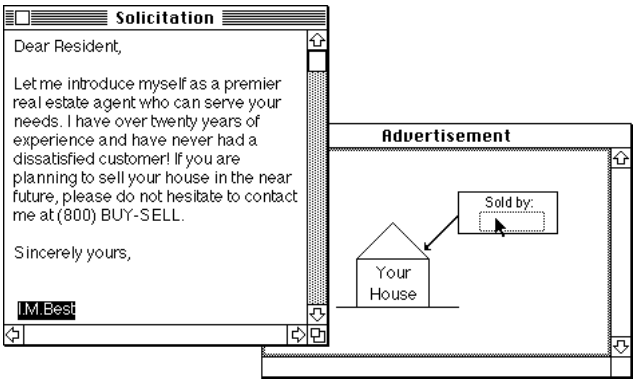
Step 2: Drag text towards another window



(Figure continued on next page)

Figure 1-2 (continued) Dragging Text from One Application to Another

Step 3: Arrive at destination



Step 4: Drop text at destination

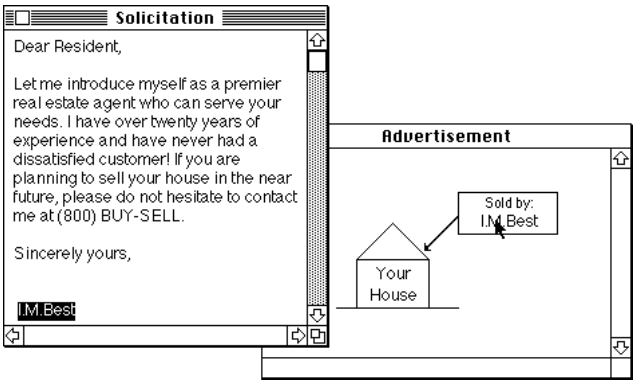
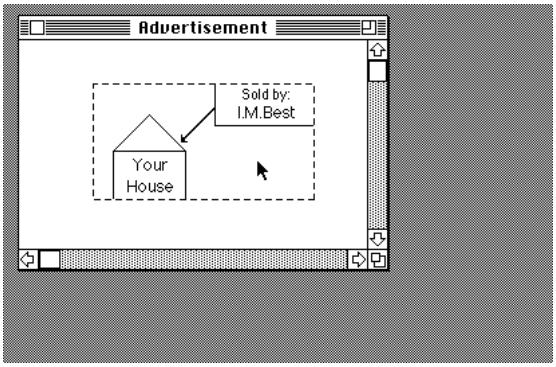


Figure 1-3 on the next page shows a situation where the source and destination may not be visible at the same time. Here, the user can create **clippings** in the Finder that store the dragged data for later use. Such clippings are created when a piece of data is dragged to a Finder window, folder, disk, or desktop. These clippings can then be dragged into another application window at a later time. See the section “Clippings” on page 18 for more details.

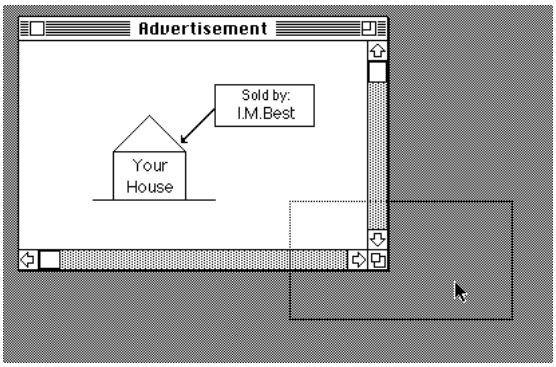
Although the Drag Manager enables easier ways of working with data, drag and drop should be considered only as an accelerator and ease-of-use technique. For every given drag-and-drop sequence, there should be another method of accomplishing the same task, such as copy-and-paste, menu items, and dialogs. Of course, there are situations where drag and drop is so intrinsic to an application that no suitable alternative methods exist. For example, dragging icons in the Finder is such a basic operation that there are no other methods of accomplishing the same task.

Figure 1-3 Dragging graphics to the Finder desktop to create clippings

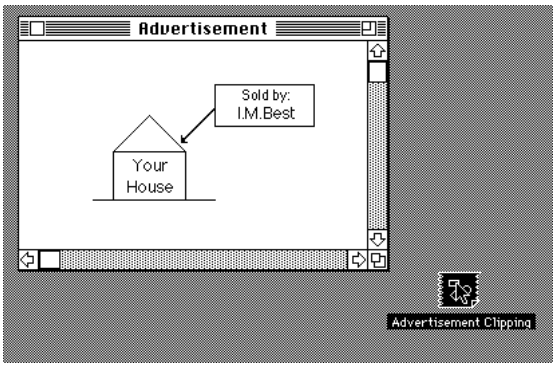
Step 1: Select graphics before dragging



Step 2: Drag graphics towards desktop



Step 3: Drop graphics on desktop



Note how the major steps of the drag-and-drop interaction model parallels a copy-and-paste sequence, where you select an item, choose the “Copy” menu command, then specify a destination, and finally choose the “Paste” menu command. However, drag and drop is a distinct technique in itself, and the Drag Manager does not use the Clipboard. Users can

take advantage of both the Clipboard and drag and drop without side effects from each other.

The user-visible elements specified by these guidelines are motivated by a single human interface principle: provide immediate feedback at the significant points during a drag-and-drop operation, as shown in the interaction model in Figure 1-1 on page 1. The guidelines are organized into sections corresponding with the following types of feedback: selection feedback, drag feedback, destination feedback, and drop feedback.

In this document, an **item** is defined to be anything that can be selected by the user, such as text, graphics, bitmaps, and icons. As the Macintosh interface uses a noun-verb paradigm, this document assumes that all items are selected before they are dragged.

Importance of Undo

As drag and drop becomes more pervasive on the Macintosh, recoverability becomes even more important. Since effecting large changes in a document is easier with drag and drop, you should extend Undo to the drag-and-drop sequences you enable in your applications.

References

The following references provide background information:

Drag Manager Programmer's Guide, by Apple Computer, September 1993.

Macintosh Human Interface Guidelines, by Apple Computer, 1992.

Selection Feedback

Since selection feedback is well-defined by the *Macintosh Human Interface Guidelines*, it is not discussed in detail here. However, there are two issues that deserve special mention: single-gesture selection and dragging, and background selections.

Single-Gesture Selection and Dragging

Since dragging is defined as moving the mouse while the mouse button is held down, a mouse-down event must take place before dragging can take place. A selection may be made as a result of this mouse-down event, just before the user starts dragging. For example, the user can select and drag a folder icon in a single gesture; the user does not have to click on the folder icon first, release the mouse button, and then click again to begin dragging the selected folder icon. Your application should ensure that implicit selection occurs (if appropriate) when the user starts dragging.

This single-gesture selection and dragging is possible only when the process of selecting an item does not require dragging. A range-selection operation, such as selecting text, a series of spreadsheet cells, or "rubber-banding" around a group of graphical objects with a marquee, do not lend to single-gesture selection and dragging because the range-selection operation itself requires dragging.

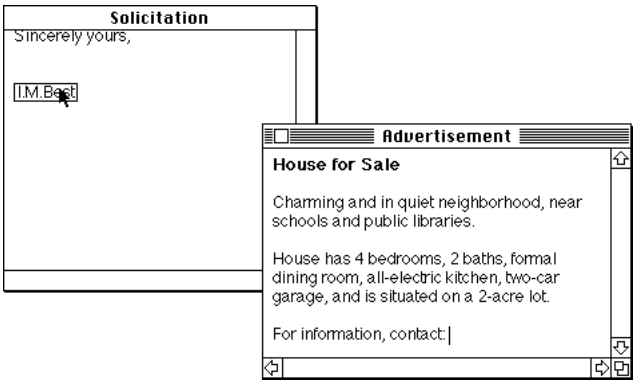
Background Selections

Traditionally, when a window containing a highlighted selection becomes inactive, the highlighting is removed, to avoid visual competition with a possible selection in the frontmost window.[1] Thus, a selection in an inactive window could only be identified by making an inactive window active. Your application can save the user from this onerous task by providing and maintaining visible **background selections**, which enable dragging of data from inactive windows to the active window.

Figure 2-1 shows an example of a background selection; note that the Solicitation window is inactive, but it contains a background selection of "I.M. Best" that can be dragged in step 2. Other examples can be found in the MPW Shell, where selections in inactive windows are identified by a single-pixel outline, and in TextEdit, which supports outline highlighting of text when the TextEdit field becomes inactive.

Figure 2-1 Dragging a Background Selection to Frontmost Window

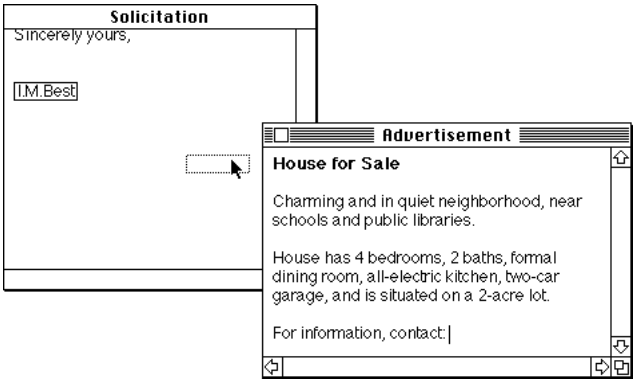
Step 1: Identify background selection



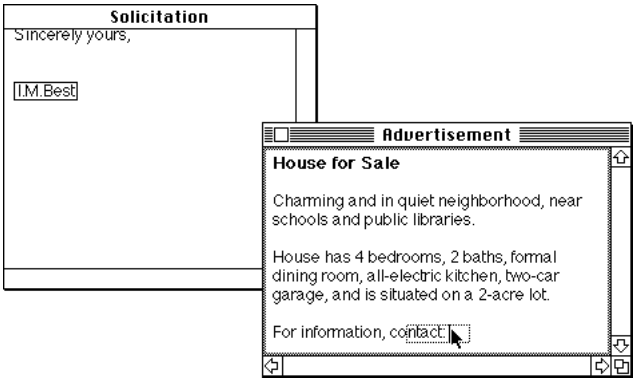
1 See *Macintosh Human Interface Guidelines*, pages 139-140 for more information on active windows.

Figure 2-1 (continued) Dragging a Background Selection to Frontmost Window

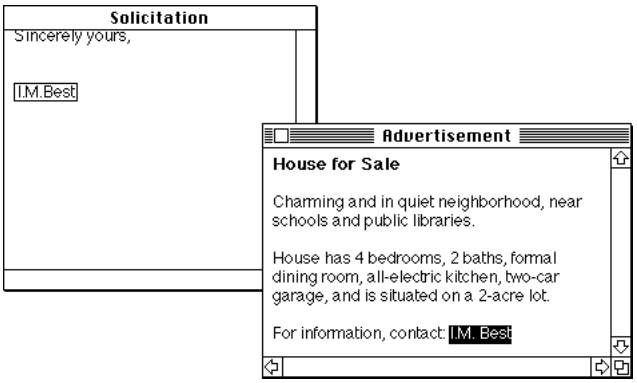
Step 2: Drag background selection



Step 3: Arrive at destination in frontmost window



Step 4: Drop text into desired location



Background selections are not required if the dragged item is discrete, such as an icon or graphical object, because implicit selection can occur when an item is dragged. However,

items selected only by range-selection operations such as text or a group of icons must have a background selection, to allow the user to drag these items out of inactive windows. Whenever an inactive window is made active, the background selection, if any, becomes highlighted as a normal selection.

Drag Feedback

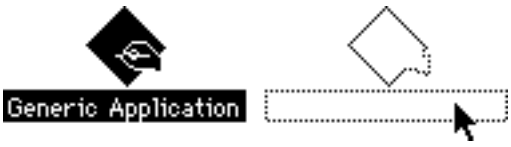
Your application provides drag feedback as soon as the user drags an item at least three pixels, which can be checked by calling a Drag Manager routine. When this occurs, your application specifies a **drag region** to the Drag Manager, which then displays the drag region at the current mouse position and tracks the mouse until the user releases the mouse button. Some drag feedback guidelines for particular types of dragged items are given in this section; these guidelines can also help in designing drag feedback for new, yet-to-be-developed types of dragged items.

In general, the drag feedback should include one or more dotted outlines, and be distinct from selection feedback. The Drag Manager generates the dotted outlines by drawing a one-pixel thick outline of the drag region with a 50% gray dithered pattern, regardless of monitor bit depth and color capability.

Icons

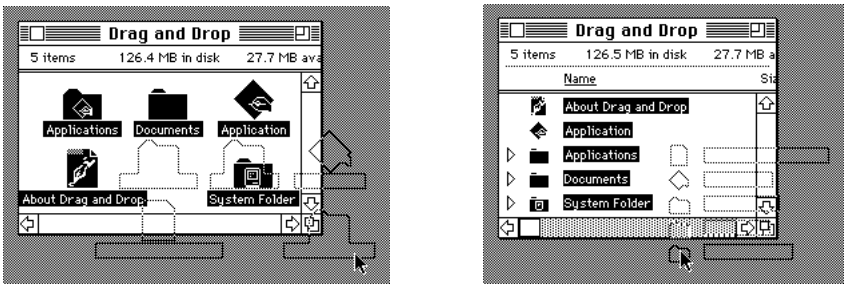
When an icon is dragged, the drag feedback should be a dotted outline that is composed of a region enclosing the icon's shape and the text label's rectangle (if there is one), shown in Figure 3-1.

Figure 3-1 Drag Feedback for a single icon



If more than one icon is dragged, the drag feedback should be a set of dotted outlines, one for each icon and text label combination. The spatial arrangement of these outlines should match that of the actual icons, as shown by the two examples in Figure 3-2. Note how each outline matches its corresponding icon's shape and text label.

Figure 3-2 Drag Feedback for two different set of icons

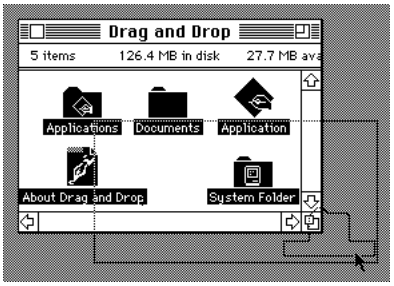


When the selection of icons is extremely large, the specification of the dotted outlines for all icons can be complex, and dragging the outline will be unwieldy. In these cases, your application can specify simplified outlines. For example, a large dotted rectangle representing the visible portion of the selection, accompanied by a full outline of the icon at the mouse position, can be used, as shown in Figure 3-3 on the next page.

Graphics

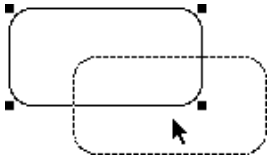
In general, when a graphical object is dragged, the drag feedback should be a dotted outline of the graphical object, as shown in Figure 3-4 on the next page. Note that the selection feedback is left undisturbed when the user drags the graphical object.

Figure 3-3 Simplified icon outlines[2]



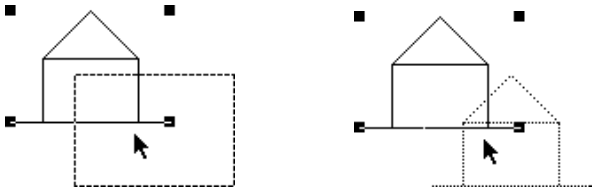
2 Don't take this figure too literally. The Finder does not exhibit this behavior in this particular situation; however, this behavior does occur when the window is very large and the number of selected icons inside the window is very large.

Figure 3-4 Drag Feedback for a graphical object



For a compound graphical object (such as a group of simple graphical objects), your application can draw a dotted outline that reflects the actual compound object as much as possible. The additional coding effort and a reasonable performance cost incurred in producing a sophisticated dotted outline is offset by a better user experience of dragging the compound object. This is especially important when a user is trying to position several compound objects with respect to one another by dragging. Figure 3-5 illustrates a simple dotted outline and a more complex one.

Figure 3-5 Comparison of dotted outlines for graphical objects



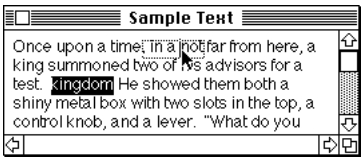
Some graphics programs, such as MacDraw II, apply a "marching ants" animated effect to the dotted outline. While this effect is not appropriate for all applications, it is useful in eliminating ambiguity when there may be graphical objects that have dotted lines.

There are some exceptions where the drag feedback for graphical objects is not a dotted outline. Notable cases are MacPaint and HyperCard; when the user drags a bitmap selection, the actual bitmap is dragged. This is called **live dragging**, and there is no drop feedback. Object-oriented graphics programs, on the other hand, move the actual object as the drop feedback, after the outline is dragged.

Text

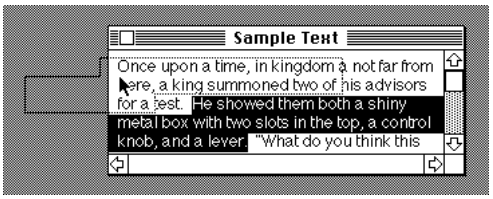
When a piece of selected text is dragged, the drag feedback should be a dotted outline of the selection, as shown in Figure 3-6.

Figure 3-6 Drag Feedback for text



Even when the text selection spans several lines, the dotted outline is based on the boundaries of the selection, as shown in Figure 3-7.

Figure 3-7 Drag Feedback of text spanning several lines



Dragging text represents a significant new capability enabled by the Drag Manager, which leads to a subtle change in how the mouse cursor behaves in text areas. Since selected text may now be dragged, the I-beam cursor cannot always appear whenever the mouse position is in a text area. If the mouse position is over a text selection that can be dragged, the arrow pointer should be shown instead. However, the insertion caret can still be set at a particular point in the selected text if the mouse-down event is immediately followed by a mouse-up event without an intervening drag gesture. In this case, the insertion caret is placed at the time of the mouse-up event, as the selection is replaced by an insertion point. Your application should support this new behavior.

Multiple Dragged Items

When several items are selected, some of these items may be outside a window's visible region. In such cases, the set of drag outlines should be limited to only those items that are visible or partly visible. When a partly-visible item is dragged, the drag outline should be generated for the entire item, to provide a cue to the user on the actual dimensions of the item.

Destination Feedback

If the user drags an item to a destination in your application, your application provides feedback that indicates whether it can accept that item as input. Destination feedback should not occur simply because your application is "drag-aware"; rather, it should depend

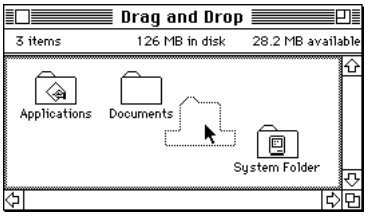
on the destination's ability to accept the type of data contained in the dragged item. For example, an editable text box that only accepts text is highlighted only when the dragged item consists of text (i.e., as opposed to a picture).

The actual appearance of destination feedback depends on the type of destination. For example, destination windows and destination folders are highlighted, but in different ways. Several windows may even have different highlighting appearances due to differences in their structures, such as Finder windows and text document windows. The Drag Manager provides some utilities for simple highlighting, while more complex highlighting must be handled by your application.

Windows

Since it does not make sense for the entire structure region of a window to be a valid destination, we use the concept of a **destination region** for windows. For example, the destination region of a document window is usually the window's content region minus the regions used for controls (such as scroll bars, size boxes, tool palettes, pattern palettes, and rulers) or informational areas. As explained later in this section, the destination region may cover a more specific area of a window, or multiple destination regions may be available in a single window. Note that window title bars or window controls are not valid destinations, thus providing "escape hatches" for the user while dragging. See Figure 4-1 for an example.

Figure 4-1 Destination Region Highlighting



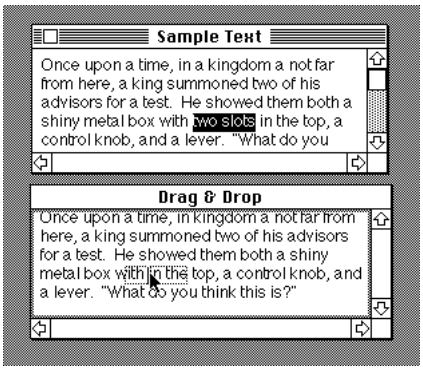
When the user drags an item from one destination region into another destination region (which may be in another window) that can accept the dragged item as input, your application highlights the destination region. The mouse pointer must be inside the destination region for it to be highlighted. As soon as the mouse pointer leaves a destination region, the highlighting for that region is removed. You can use the Drag Manager to specify your destination regions; the feedback drawn by the Drag Manager consists of a two-pixel rectangular frame that matches the size of the destination region. On black-and-white monitors, this frame is drawn using a 50% gray pattern, and on color monitors, the Drag Manager draws this frame in a color based on the window color specified in the Colors control panel. In the case of color, the frame is drawn in Color QuickDraw's `hilite` mode.

If a drag-and-drop sequence takes place wholly within a single destination region (such as moving a document icon to a different location in the same folder window), your application should not highlight that destination region, to avoid distracting the user.

However, if the user drags an item completely outside of a destination region, and decides to drop the same item back in the same destination region, the destination region should be highlighted, providing positive feedback that the dragged item can be released inside the destination region where it came from.

Destination region highlighting can be supplemented by additional feedback. An example is insertion point feedback when dragging text from one window to another text window, as shown in Figure 4-2, and explained in more detail later in this section. Note that insertion point feedback is provided even though the window is inactive.

Figure 4-2 Destination Region Highlighting and Insertion Point Feedback

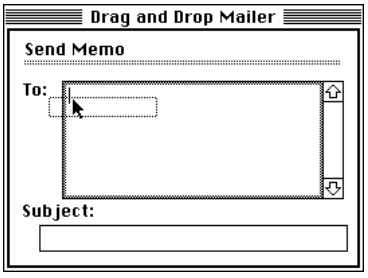


There are situations where highlighting a more narrowly-defined area of a window is more appropriate than highlighting the entire content region; examples are spreadsheets, text boxes, fill-in forms, and panes. In these cases, the destination region must be tailored to more precisely indicate the specific destination. Figures 4-3a-c on the next page show several examples of customized destination regions and highlighting. Figure 4-3a is an example of a mailer application window with two text boxes; Figure 4-3b is an example of a spreadsheet that uses the “current cell” frame to indicate the destination cell; and Figure 4-3c is an example of a list that uses a two-pixel solid black line between cells to indicate the destination. In the first two examples, there are multiple destination regions, but only one destination region is highlighted at a time.

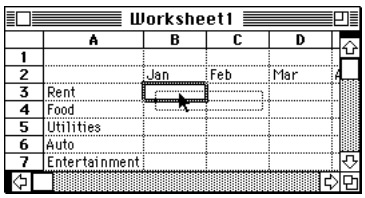
Note how the insertion point in the “To:” text box in Figure 4-3a is also a part of the destination feedback. Also, the two-pixel solid black line between cells in Figure 4-3c makes sense when the user can control the ordering of the items in the list. If your application imposes an ordering (such as alphabetic order) on a list, the two-pixel solid black line should appear at the location where the dropped item would be inserted, if that location is visible. If that location is not visible, only the destination region highlighting is applied to the list box.

Figure 4-3 Special Types of Destination Region Highlighting

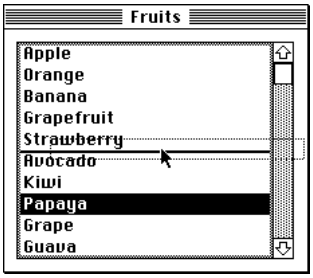
4-3a: Text Box Highlighting



4-3b: Spreadsheet Cell Highlighting



4-3c: List Highlighting



Icons

When the user drags an item to an icon, such as one representing a folder, application, and hard disk, the icon is highlighted. On black-and-white monitors, the icon's bitmap and its text label (if there is one) are inverted, as in Figure 4-4. On gray-scale and color monitors, icons are highlighted by darkening the color of the bitmap and inverting its text label, as exhibited by the Finder in versions 7.0 and later.

Figure 4-4 Icon Highlighting

Graphics

Graphical objects are different from icons in that they are not containers. Graphics typically do not have a mandatory hierarchy like that of folders; they usually share a "canvas" of

space (but may be grouped). Accordingly, destination feedback is limited to highlighting the destination region of the window if it is different from the source window.

Text

While the user is dragging an item to a text area, the destination feedback should be a solid vertical line at the point in the text where the dragged item would be inserted if the user releases the mouse button. This vertical line is identical to the insertion caret used by TextEdit. If the user pauses while dragging, the vertical line starts blinking, using the rate of insertion point blinking specified in the General Controls control panel. The vertical line is usually accompanied by other feedback such as destination region highlighting.

Multiple Dragged Items

If there are multiple dragged items, the destination feedback should occur only if it can accept all of the dragged items. The reason for this is that a set of dragged items may be heterogeneous, and the destination may not be able to accept all types contained in that set. If the destination cannot accept all of the dragged items, destination feedback does not occur, and the user's attempt to drop the set on such a destination results in abort feedback. An optional behavior would be to let the user know of the "offending" type in a dragged set if the user repeatedly tries to drop the set on a destination.

In the case that the destination can accept all of the dragged items, the destination should accept these items in the order specified by the source. The source application should organize the dragged items in the order in which they were selected, except for two cases. If the dragged items come from ordered views (such as View by Date, or alphabetized lists), that view's ordering takes precedence over the selection order. If both the source and destination provide a spatial ordering (such as graphic applications), the spatial ordering takes precedence over the selection order.

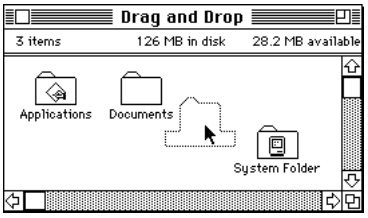
Autoscrolling

When an item is being dragged, your application must determine whether to autoscroll or allow the item to "escape" the window. If your application allows items to be dragged outside of windows, you should define an autoscrolling region, as shown in Figure 4-5. While dragging, if the mouse position enters and remains in the autoscrolling region for at least 10 ticks, autoscrolling takes place until the mouse position exits the autoscrolling region. Thus, if the mouse position is completely outside the window (or in the title bar), autoscrolling is suspended, allowing the dragged item to escape the window and be dropped at destinations outside the window. If the mouse enters one of the four corner blocks of the autoscrolling region, autoscrolling takes place in both horizontal and vertical directions simultaneously.

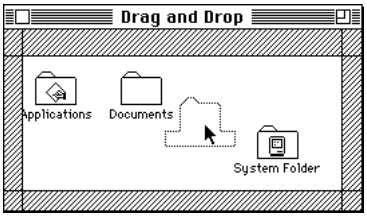
Autoscrolling of a destination window takes place only if the window is also the source window and is frontmost. You should not autoscroll inactive windows.

Figure 4-5 Narrowly-defined Autoscrolling Region

Typical Finder Window



Autoscrolling Region



Drop Feedback

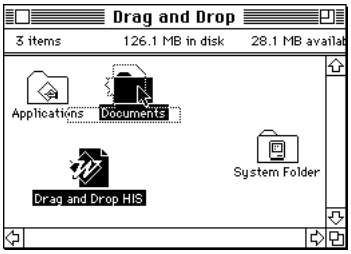
When the user releases the mouse button after dragging an item to a destination, there should be some sort of drop feedback, informing the user of the success of the drag-and-drop operation. While this feedback can be visual, it is primarily behavioral in nature. The behavior comes from the semantic operation indicated by the drag-and-drop sequence. Examples of this behavior are given below.

Finder Icons

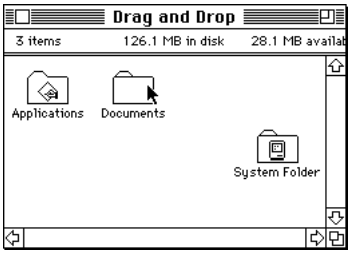
When the user drags a document to a folder icon in the Finder, the behavior of the drop feedback is the reorganization of the document into the folder; the visual component is the disappearance of the document icon and the unhighlighting of the destination folder icon (in the case of a move operation). An example is shown in Figure 5-1.

Figure 5-1 Dragging a Document to Folder

Drag to Folder



Drop into Folder



If an icon represents a system service, such as a mailbox or printer, the drop feedback should be followed by some indication that the service is being delivered. For example, if

the user drags a document to a printer icon, the icon might slowly "fill up" in color as the printing job progresses towards completion. This is called **progress feedback**.

Graphics

When dropping graphics, the drop feedback is usually the movement of the actual item to the location of the mouse-up event.

Text

After dropping text, the drop feedback is the movement or copying of text from the source to the destination. If a **move** operation is in effect, the source text disappears. In either case, the text inserted at the destination is selected. When moving text in the same window, a series of "zooming rectangles" from the source text to the destination text should be displayed after the text is rewrapped, as the destination text may end up at a different point than the exact point where the user dropped the text (due to the rewrapping). Thus, the zooming rectangles provide an important user cue.

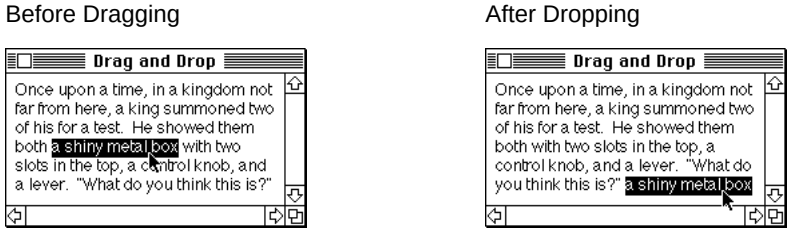
Since text is being inserted at the destination, intelligent cut-and-paste rules, as explained on pages 301-302 of the *Macintosh Human Interface Guidelines*, should be supported.

When text is dropped in a destination that supports styled text, the font, face, and size attributes of the source text should be applied to the dropped text. If the destination does not support styled text, the dropped text should take on the current font, face, and size attributes as indicated by the insertion point at the destination.

Transferring Selections

After a successful drag-and-drop sequence involving a single window, the selection feedback is transferred from the source to the destination. This behavior provides an important user cue and allows the user to reposition the selection without having to make the selection again. Figure 5-2 shows an example.

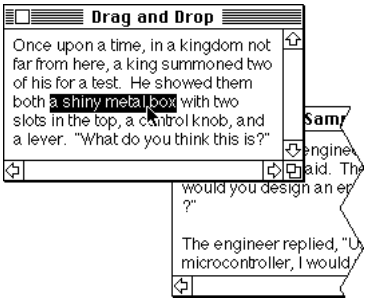
Figure 5-2 Transferring Selection Feedback



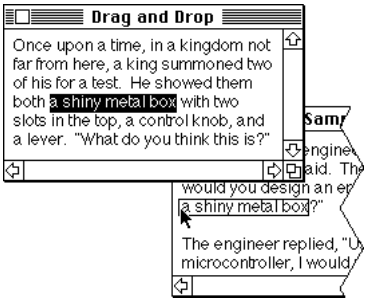
If the drag-and-drop sequence involves two windows (one active and one inactive), the concept of background selection comes into play. If the user drags an item from an active window to an inactive window, the dragged item becomes a background selection at the destination; the selection in the active window remains selected. These guidelines also apply in the reverse situation where an item is dragged from an inactive window to an active window. Figure 5-3 on the next page shows a situation where a piece of text is dragged to the position just before the question mark in the inactive window.

Figure 5-3 Updating Selection Feedback in Multiple Windows

Before Dragging to Inactive Window



After Dropping in Inactive Window



Note that the inactive window does not become active as a result of the drag-and-drop sequence; see the section "When to Activate Inactive Windows" below for more details. Also, when content is dropped into a window, your application must be careful not to overwrite the window's selection. Rather, the selection should be deselected before the content is pasted.

By the way, to make it easy for the user to add text after dropping a text selection, note that when a block of text is selected, pressing an arrow key deselects the range. If the Left Arrow key is pressed, the insertion point goes to the beginning of what had been the selection. If the Right Arrow key is pressed, the insertion point goes to the end of what had been the selection. See *Macintosh Human Interface Guidelines*, page 296.

Confirmation Dialogs

At drop time, a confirmation dialog may be presented to the user if the action resulting from the drag-and-drop sequence is unrecoverable. For example, dropping an icon into a write-only "drop box" on a shared volume is unrecoverable because the user does not have sufficient privileges to open the drop box and undo the action. In this case, a confirmation dialog should be presented to the user. Similarly, dropping a document on a AOCE business card invokes a dialog asking the user to confirm a "send" action, since that action is unrecoverable, once executed.

Abort Feedback

Dropping outside of an acceptable destination is considered as an "abort" and should be indicated by zooming rectangles that originate at the last position of the drag feedback and end at the source's location. (This is also called a "zoomback.") For example, dragging an item to the menu bar (which is not a valid destination) and dropping it there results in abort feedback. Also, dropping an item on window title bars or window controls (such as scroll bars or size boxes) results in abort feedback, as they are not valid destinations. The Drag Manager provides this feedback when it determines that no receiver requested the sender's information.

If, for some reason, dropping inside a destination does not result in a successful operation (for example, due to insufficient memory to handle the added item), zooming rectangles should also be used. This is a form of **negative** drop feedback.

When to Activate Inactive Windows

Since many drag-and-drop sequences involve more than one window, certain rules regarding the activation of windows must be observed for reasons of consistency and streamlining work flow.

Historically, an inactive window becomes active upon a mouse-**down** event in that window. Since the user may drag an item from an inactive window to the frontmost window, and continue working in the frontmost window, this behavior must be slightly modified so that, in a certain case described below, the mouse-**up** event serves as the window-activation trigger, rather than the mouse-**down** event.

The only situation where a mouse-**up** event, instead of a mouse-**down** event, activates an inactive window occurs when the user presses the mouse button on a draggable item (such as a background selection, icon or graphic) in an inactive window and releases the mouse button in the **same** window. In this situation, the inactive window becomes active as soon as the mouse button is released. (Note that the user does not necessarily have to drag the mouse between the mouse-**down** and mouse-**up** events.) If the mouse-**down** event does not occur on a draggable mouse item, inactive windows become active upon that event, just as before.

NOTE
To allow users to drag an item from an inactive window without activating that window, your application must support **click-through**. Otherwise, pressing the mouse button would activate the inactive window immediately, and the user would have to press the mouse button again to select a draggable item. ▲

Because of the above rule, when the user drags from one window to another window that is inactive, the inactive destination window is not activated at drop time. If the drag-and-drop sequence ends at the Finder desktop, no window is brought to the front. Remember that

whenever an inactive window becomes active, the background selection (if any) in that window becomes highlighted as a normal selection.

Clippings

When an item is dragged from an application to the desktop, the Finder creates a **clipping** that contains the data in the dragged item. These clippings are similar to sound, font, and edition files in that they are not owned by any particular application. If discontinuous selections are dragged from a source to the Finder, a separate clipping is created for each item in the selection. The Finder icons for clippings can be either thumbnails (small pictures that resemble the dragged data), or system-defined icons for TEXT and PICT; these icons do not look like document icons. The system-defined icons are shown in Figure 8-1 on the next page.

Figure 8-1 System-defined icons for clippings

Your application should provide a number of representations (such as TEXT, PICT, and native formats) to ensure flexibility with different destinations. Regardless of which representations are stored, **round-trip data integrity** should be preserved; a clipping dragged back into its source should be identical to the original item. When your application receives a clipping, it should use the richest representation supported, and store the other representations, if possible. Storing all the representations of a clipping inside documents allows **scenic-route data integrity**; dragging the same clipping from a source through several destinations and back to the source should result in no fidelity loss.

When clippings are created, each clipping is given a default name, which is a concatenation of the type of data was dragged, the word “clipping”, and, if necessary, a number that disambiguates the entire name in the destination Finder window or desktop. For example, dragging some text from a document to the Finder would generate a clipping name of “text clipping”. A subsequently-created clipping placed in the same destination would result in a clipping name of “text clipping 2” to avoid a naming conflict. If the type is unknown, it is omitted from the clipping name.

The user can open clippings in the Finder and view a representation of the data in a modeless window, similar to the Clipboard window. The user cannot select, copy, or edit any of the contents in these windows. If a PICT or TEXT representation is stored in the clipping, the Finder displays that representation. Otherwise, the string “This clipping has no text, picture, or sound.” is shown in the window.

Drag and Drop Semantics

Move *vs.* Copy

When drag-and-drop sequences take place, your application needs to determine whether to **move** or **copy** the dragged item after it is dropped on a destination. The actual behavior depends on the context of the drag-and-drop operation, as described below.

In general, if the source and destination are in the same container (e.g., a window or a volume), a drag-and-drop sequence is interpreted as a move operation (i.e., cut-and-paste). To specify a copy operation within the same container, the Option key is used when dragging. If two different containers are involved, a copy operation (i.e., copy-and-paste) is executed as the default action.

You cannot assume that a window is always a container; you must consider the underlying data structure of the contents in the window. For example, if your application allows two windows to display the same document (i.e., provides multiple views of the same data), a drag-and-drop sequence between these two windows should be a **move** operation. For example, the Finder uses the concept of a volume as a container in determining its semantics. If the two windows belong to the same volume, the Finder performs a move operation, and if two different volumes are involved, a copy operation is performed. However, a copy operation between two windows of the same volume is specified by holding down the Option key at drop time. Also, desktop services (such as mailers, printers, or converters) are not necessarily containers. Since a desktop service performs a transformation or an action on the dropped item without affecting the original, dragging to a desktop service always implies a copy operation.

There are also cases where it does not make sense to move or copy data or containers. For example, AOCE provides directories as a type of network container. Attempting to drag the network container to your desktop would imply copying all the subdirectories and contents—a potentially expensive operation. Rather, AOCE makes an alias to that container on your desktop, so that you can have ready access to a network container.

When to Check the Option Key

There are two timing rules for checking the state of the Option key in determining whether to perform a copy operation; one rule is mandatory, and the second is optional. The first rule is to check if the Option key is held down just before starting a drag-and-drop sequence. In this case, the semantics of the operation (i.e., copy) are established at the beginning and remain in effect during the entire drag-and-drop operation, even if the user releases the Option key before dropping the dragged item. This behavior is exhibited by many bitmap graphics programs such as Claris' MacPaint. This rule should be supported by all applications.

The second rule is to check if the Option key is held down at drop time only if the user did not hold down the Option key at the beginning of the operation. If your application

supports this rule, the user has the flexibility of making a move *vs.* copy decision at a later point in the drag-and-drop sequence. Unlike the first case described above, pressing the Option key *during* the drag-and-drop sequence does not "latch" for the remainder of the sequence; the state of the Option key is checked only at drop time.

Consistent Semantics of the Option Key

The Option key does not act as a toggle switch; Option-dragging between containers still means a **copy** operation (albeit redundantly). This guideline allows users to learn that Option means Copy.

Guiding Principle and Common Contexts

The principle driving the above guidelines is to prevent the user from accidental data loss. Moving data across applications may result in potential data loss because an Undo command in the destination application does not trigger an Undo in the source application. Moving data within the same window (or same volume, as in the case of the Finder) does not lead to data loss because the same memory structure is involved.

Table 9-1 on the next page lists several common contexts and the semantics of each context.

Table 9-1 Common Contexts and Drag-and-Drop Semantics

Drag data in a document to a place inside the same document	Move
Drag data in a document to a place in another document	Copy
Drag data in a document to any location in the Finder	Copy*
Drag icon from Finder to any document	Copy
Drag icon from Finder to another place on same volume	Move
Drag icon from Finder to any place on another volume	Copy
Drag data from document to a desktop service	Copy
Drag icon from Finder to a desktop service	Copy

Ambiguity

You should avoid situations where more than one action is applicable for a specific drag-and-drop sequence. This ambiguity and overloading usually leads to some user negotiation that can be confusing and intrusive.

* In this case, a clipping is created by the Finder.

Using the Trash as a Destination

As interapplication dragging becomes possible, the Trash becomes a more useful destination. The Drag Manager makes the Trash available to applications.

Dragging items to the Trash results in **moving** the item from the source to the Trash. For example, dragging a text selection from a word-processing application and dropping it on the Trash icon (or in the Trash window) results in the text being deleted from the application and a clipping containing that text being created inside the Trash. Note that the item is moved, although it is dragged between two containers. This exception to the semantics described above is appropriate because the user can undo the operation by dragging the clipping out of the Trash back to its original source; it is consistent with the principle of preventing accidental data loss.

It is important to preserve the Trash's container property; do not simply delete the source without creating a clipping or other item in the Trash.

Modifier Keys

As mentioned earlier, the Option key is used for copying while dragging. The Shift-drag and Command-drag gestures are for use by your application. The Control-drag gesture is reserved for future use by Apple.

Checklist

This checklist contains questions about the drag-and-drop interface that you can ask yourself while reviewing your software. These questions will help bring to mind the particulars of the guidelines.

You must be able to every question "yes" to ensure conformity with the guidelines. However, sometimes it is necessary to make tradeoffs in your application in order to make the most usable interface. Not all guidelines may apply to your particular application, but remember to maintain the spirit of these guidelines.

General

- Do you provide alternative methods for accomplishing a given drag-and-drop sequence (such as copy-and-paste or other menu items)?

Selections

- When the mouse-down event occurs at the beginning of a drag-and-drop sequence, do you check to see if the item at the mouse position is already selected? If the item is not already selected, does it become selected before dragging takes place? (This behavior is identical to that of the Finder.)

- If you allow dragging from inactive windows, do you ensure that highlighted selections in an active window become background selections when the window becomes inactive?

- When an inactive window containing a background selection is brought to the front, does the background selection become highlighted as a normal selection?

Dragging

- When an item is dragged, do you provide drag feedback? Does the drag feedback contain a dotted outline reflecting the size and shape of the item?

- When more than one item is dragged, does the drag feedback contain a set of dotted outlines (one for each visible or partly-visible item), with the same spatial organization as the actual items?

- Are dotted outlines generated by drawing a one-pixel thick outline of the drag region with a 50% gray dithered pattern?

- When the mouse pointer is over a draggable text selection, does it change into an arrow shape to indicate that the text selection can be dragged?
- When dragging an item towards the edges of a scrollable window, does your application determine whether to autoscroll and allow the item to escape the window?

Destinations

- When the user drags an item over a destination, do you provide destination feedback?
- Does the destination feedback occur only when the destination can accept the type of data contained in the dragged item? When there are multiple dragged items, does the destination feedback occur only when the destination can accept all of the dragged items?
- When dragging an item into a text area, does the destination feedback include a solid vertical line (identical to that of TextEdit) at the point in the text where the mouse pointer is located? If the user pauses while dragging, does the vertical line start blinking?
- Are destination regions for windows defined appropriately? Is a destination region highlighted only when the mouse position is inside that region? If you provide multiple destination regions in a window, do you ensure that no more than one destination region is highlighted at a time?
- Is a destination region highlighted when the user drags an item outside of that destination region and decides to drop the same item back in the same destination region?

Dropping

- When the user drops an item over a destination, do you provide drop feedback (which is primarily behavioral in nature)?
- When moving text in the same window as the source , are “zooming rectangles” drawn from the source text to the destination text (after rewrapping text)? Are intelligent cut-and-paste rules supported? Are font, size, and style attributes preserved across drags when possible?
- After a successful drag-and-drop sequence in a single window, is the selection feedback transferred from the source to the destination? If the drag-and-drop sequence involves an inactive window, does the dragged item become a background selection in the destination window?
- Is a confirmation dialog presented at drop time if the impending action is unrecoverable?

- When the user drops multiple items, are the items accepted in the order in which they were selected, except when priority-ordered or spatially-ordered?

Aborting

- If the user drops a dragged item outside a destination (such as the menu bar, window title bars, controls, etc.), are zooming rectangles drawn from the dragged outline back to the source?

- If dropping on a destination does not result in a successful operation, are zooming rectangles drawn from the dragged outline drawn back to the source?

Windows

- Are inactive windows activated when the source and destination of a drag-and-drop operation are in the same window or when the user presses the mouse button in non-draggable regions?

Semantics

- Is a drag-and-drop sequence within the same container interpreted as a move operation? Is a drag-and-drop sequence across two containers interpreted as a copy operation?

- Does the Option key indicate a copy operation when dragging within the same container or across two containers? Do you check the state of the Option key at mouse-down time? When appropriate, do you also check the state of the Option key at mouse-up time, which allows more flexibility during drag-and-drop sequences?

- When the user drags an item from your application to the Trash, do you provide the item to the Trash and delete the item from the source?

- Do you avoid situations where more than one verb is applicable for a specific drag-and-drop sequence?