Making It Macintosh: An Interactive Human Interface Instructional Product for Software Developers

Harry J. Saddler

Developer Technical Publications Group
Apple Computer, Inc.
20400 Steven Creek Blvd., MS 75-5A
Cupertino, CA 95014

ABSTRACT

Making It Macintosh: The Macintosh Human Interface Guidelines Companion is an interactive instructional product designed and developed by Apple Computer, Inc. *Making It Macintosh* uses computer-based animation and interaction to document the Macintosh user interface, illustrate human interface design issues, and provide interface implementation strategies for software developers. This paper describes the product's audience, its goals, its design, and the specific techniques used to present its content to the user.

AUDIENCE

The audience of *Making It Macintosh* is composed primarily of programmers developing software products for the Macintosh platform. Additional audiences include human interface designers, software product managers, and educators and trainers in the field of human interface design.

GOALS

The primary goals of *Making It Macintosh* are

- To provide a standard of good human interface design for software developers;
- To educate developers about what constitutes good (and bad) interface design;
- To provide the resources necessary for developers to create good human interfaces;
- To motivate developers to care about human-computer interface design.

DESIGN

A principal consideration in the design of *Making It Macintosh* was the appropriate application of interactive technologies to the task of presenting human interface issues. *Making It Macintosh* was designed as a companion to Apple's book *Macintosh Human Interface Guidelines* (Addison-Wesley, 1992), which comprehensively documents the Macintosh human interface and provides interface design guidance.

Interactive media adds value to the *Guidelines* by

- Dynamic depiction of interface behavior and user-system interactions;
- Depiction of successive improvement in interface problem simulations;
- Non-literal, conceptual depictions of interface elements and behaviors;
- Compelling comparison of design alternatives.

INSTRUCTIONAL TECHNIQUES

Human interface issues to be presented are composed into separate instructional "frames" or screens (Figure 1). Each frame contains one or more animated segments. Each segment is represented by a short textual narrative and an on-screen button that triggers the animation segment.

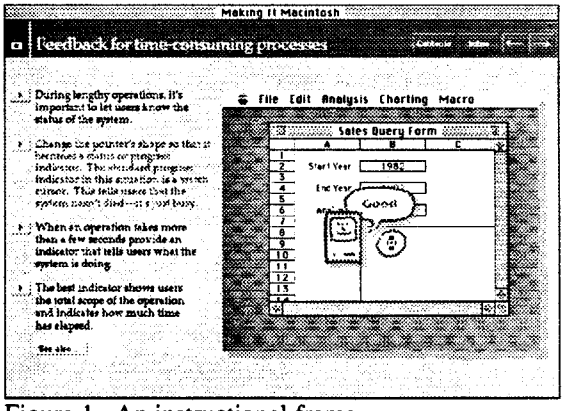

Figure 1. An instructional frame

The instructional mode of each frame generally falls into one of four categories:

- Standard interface element appearance, behavior, and use;
- Comparison of interface elements as potential interface design solutions;
- Demonstration of interface design problems and potential solutions;
- Non-literal depiction of interface elements and interactions by adding a dimension (time or simulated third spatial dimension).

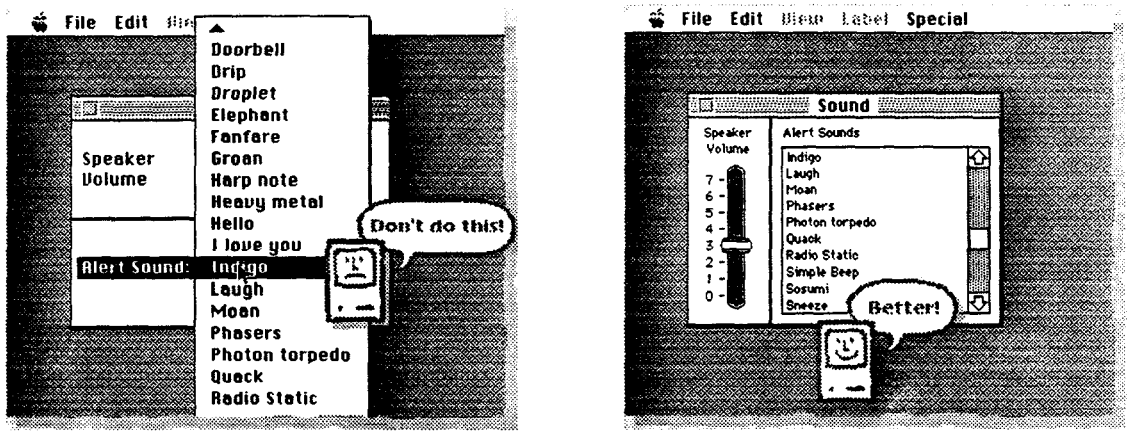

Figure 2. An interface problem, and a suggested solution.

The animations generally depict Macintosh interface elements as they are manipulated by a user. In other cases, interface elements are juxtaposed for comparison, or transfigured to demonstrate an improvement or the selection of a more appropriate interface element or style.

The interface depictions are annotated with various callouts, labels, and commentators. Figure 2 shows an animated commentator (known to the development team as "Luke") who provides evaluation of alternative design solutions. The use of a humorous graphic character with a colloquial patois is an attempt to find a way to point out design flaws without evoking the "interface police" characterization of interface guidelines.

The ostensible user occasionally makes an appearance as a commentator, with his or her thoughts appearing as text in a cartoon "thought balloon". This device is useful when the effect of a poor interface design is user confusion and frustration. It is admittedly a simplistic representation of the user experience; a more compelling version might, through audio and/or video, portray the "user" expressing their frustration (or satisfaction) with the interface.

These annotations are visually designed to appear as though occupying a layer atop the interface depictions, which, in turn, occupy a layer atop the product's own interface. This is necessary to help the user distinguish between the *depicted* interface and the *product* interface, both of which comprise standard Macintosh interface elements and styles.

The depicted interface elements are not interactive; rather, they are graphical animations, the pace and order of which are controlled by the user. This "indirect interaction" greatly simplifies the instructional design process. The need for fidelity in the interface elements' appearance and behavior is limited by the context of the instruction.

If the depicted interface elements were directly manipulable by the user, we felt it would be necessary to implement the full (or at least a greater) range of possible interactions.

Of course, in many cases, interacting directly with an interface element would be a more effective way to understand its behavior. We may, in future versions, link these animations to working sample applications, allowing the user to manipulate working interface elements to reinforce the instructional material.

FUTURE WORK

The current product version contains 100 individual instructional frames.These document the *artifacts* of the human interface; future work might focus on helping the software developer understand and apply the *processes* of human interface design. For example, interactive, self-paced tutorials in practical interface design skills such as dialog design, icon design, and control layout would be a natural complement to the more fine-grained, reference style instructional materials that are described here.

Creating interactive instructional content at several different levels of the human interface design enterprise — from source code to the principles of interaction design — should yield an additional benefit. Software developers faced with specific implementation guidelines can ask "Why?", and the product can answer with more abstract, conceptual instruction, perhaps citing (and demonstating) a principle of human-computer interaction. Similarly, principles treated in an abstract, conceptual way can be supported by the implementation-specific, real-world examples. The result would be a richly textured compendium of practical facts and usable processes that can help software developers in the day-to-day development of usable applications.