The Developer Business Report *January 1994*

Apple Directions

Inside This Issue

Apple News

Planned Quadra Model Runs DOS/Windows, Wins Award

At Fall Comdex in Las Vegas last month, Apple announced a technology that will allow a planned new model—the Macintosh Quadra 610, DOS Compatible version computer—to run MS-DOS and DOS software, all at a price expected to add less than $500 to the price of the basic computer. Apple introduced this technology at Comdex as a sign of its commitment to selling to the DOS market. Comdex evidently took notice—interest in the Macintosh Quadra 610, DOS Compatible version computer was high at the Apple booth, and the computer won the show's "Best [New] System" award.

Apple has created a DOS computer on a processor-direct slot (PDS) card that currently fits in a Macintosh Quadra 610 computer. As a stand-alone product, it is expected to retail for under $500 (going under the name of the DOS Compatibility Card). The Macintosh Quadra 610, DOS Compatible version computer (a Macintosh Quadra 610 with the PDS card) should add somewhat less than $500 to the price of the basic computer. Apple is aggressively pursuing the implementation of this technology and will have an update on

Please turn to page 7

Strategy Mosaic

QuickDraw GX: It's Your Type—Exactly

By Gregg Williams, Apple Directions *Staff*

Type is, to our minds, like air is to our bodies—it's invisible, there's not (for most people) much to say on the subject, but you can't live without it. Electronic publishing (and the Macintosh computer and LaserWriter printer in particular) brought sophisticated typography into wider use than ever before. But when type went electronic, we lost some of the graphical expressiveness of traditional type. The reason for the loss was understandable enough: Typography is complicated, and the technology of the day couldn't implement everything. What it did offer was more than enough, though, and people were glad to have it.

That changes *now.* You've heard a lot about QuickDraw GX (see the November 1993 Developer CD and "Rethinking Your Applications for QuickDraw GX," *Apple Directions,* October 1993), but you may not know how it's going to revolutionize desktop—and therefore mainstream—publishing. QuickDraw GX greatly increases the level of control you have over fonts and font styles, so much so that QuickDraw GX makes it easy to do almost anything that's ever been done using traditional type—and more.

Please turn to page 4

AppleDirections

Volume 2, Number 1

Apple Directions, Apple's monthly developer newsletter, communicates Apple's strategic, business, and technical directions to decision makers at development companies to help maximize their development dollar. It is published by the Developer Support Information group within Apple's Developer Press.

Editor
Paul Dreyfus (AppleLink: DREYFUS.P)

Technical Editor
Gregg Williams (GREGGW)

Business & Marketing Editor
Dee Kiamy (KIAMY)

Designer
Robert Stone

Production Editor
Lisa Ferdinandsen (LISAFERD)

Contributors
Meredith Best, Juan Bettaglio, Pete Bickford, Alex Dosher, Tim Enwall, Amanda Hixson, Stacy Moore, Caroline Rose, Anne Szabla, Jessa Vartanian

Manager, Developer Press
Dennis Matthews

Manager, Developer Support Information
Greg Joswiak

Production Manager
Diane Wilcox

PrePress/Film
Aptos Post

Printer
Wolfer Press Co., Inc., Los Angeles, CA

Editor's Note

Year-End Close

1993 finished with a flurry of Apple news, so much so that I need to use this space to tell you about a few items that we couldn't cram in elsewhere. So, without further ado . . .

Apple Enters Preschool Market

Recognizing the fascination preschoolers have with computing and the needs of preschool educators, Apple Computer, Inc., has bundled its CD-ROM drive-equipped Macintosh LC 520 computer with early childhood development software and CD-ROM titles into the Early Childhood Connections Package. This is Apple's first-ever product offering for the U.S. preschool market of some 18 million children.

If you have products or ideas that you think might be appropriate for bundling with future versions of the connections package, send an AppleLink message to Maurilio Flores at FLORES.M.

New Remote Access Products

Apple recently announced a new family of products that give users remote access to AppleTalk network services over telephone connections. The new Apple Remote Access (ARA) line includes the following:

- ARA MultiPort Server, a serial card that converts a Macintosh computer into a server that supports up to 16 multiple dial-up connections simultaneously, giving users remote access to the network services they use when they're in the office.
- ARA Personal Server for Macintosh, which replaces AppleTalk Remote Access 1.0. Users install server software on office computers connected to an AppleTalk network; after installing client software on their home computers, they can dial in to their office computers and access the office network.
- ARA Client for Macintosh, the client software just mentioned that gives Macintosh and PowerBook computers remote access to their office systems.

The new products open opportunities for you to build remote access servers, provide products that enhance security, and develop applications that track and report network activity. You can also customize your applications so that customers can use them to access ARA services directly. You can purchase the ARA Protocol Developer's Kit, version 2.0, from APDA (# AAAAAAA).

FireWire Wins Best of Comdex Award

The DOS-compatible Macintosh Quadra 610 wasn't the only Apple technology to win a *Byte* magazine Best of Comdex award. *Byte*'s Most Significant Technology award went to Apple and Texas Instruments (TI) for the development of FireWire, their implementation of the P1394 high-speed serial bus. TI and NCR have announced plans to develop chips based on the technology.

FireWire technology is intended to anticipate the needs of future systems for transport of large volumes of data in an isochronous, or "real-time," manner. It will enable higher bit rates and lower cable costs than currently possible, improving the performance of peripheral devices without increasing costs. Integrated circuits employing FireWire will enable computers to exchange information at very high speeds with a variety of electronic equipment through miniaturized connectors and low-cost cables.

That's it for the 1993 news; thanks for all your hard work this year. Now, it's on to what we think will be a very exciting New Year for the Macintosh computer.

Paul Dreyfus
Editor

Apple Directions On Line—February

The February issue of *Apple Directions* will be available on AppleLink as follows:

January 1—Preliminary draft copy
January 15—Final copy

To view the February issue of *Apple Directions* on line, follow the AppleLink path Developer Support:Developer Services:Periodicals:Apple Directions:Apple Directions February. ♣

IndustryWatch: News & Perspective

Compton's Erects a Toll Booth on the Digital Highway

By Amanda Hixson, Consultant Instant Insight

According to the Second College Edition of the *American Heritage Dictionary,* published by Houghton Mifflen (copyright 1991), the term *multimedia* is defined as follows:

Including or involving the use of several media: a multimedia display.

If any of you remember the CD-ROM conferences sponsored by Microsoft in the mid-eighties, you might recall the Microsoft Press book entitled *The New Papyrus,* published in 1986. It contains numerous references to multimedia and does so in conjunction with CD-ROM discs and the access of large, diverse databases of information using various retrieval methodologies and algorithms.

Yet on August 31, 1993, the U.S. Patent Office granted Compton's New Media, Inc. (now owned by Tribune Co. of Chicago) of Carlsbad, CA, patent number 5,241,671, titled "Multimedia search systems using a plurality of entry path means which indicate interrelatedness of information." Much like the term *multimedia,* this equally vague title encompasses a patent that—according to Stanley Frank, Compton's president, as quoted in the *San Jose Mercury News*—is clear evidence that "we [Compton's] invented multimedia."

Given that the original patent was filed in October of 1989, and, as noted earlier, the term *multimedia* predates the patent application by a minimum of three years, I'm not sure how valid Compton's claim of multimedia invention is. But I've got to admit that Mr. Frank has an extremely large set of encyclopedias to make such a claim with a straight face.

The news release I received by fax from Pat Meier Associates for Compton's contained additional information explaining the key points of the patent to those of us without degrees in patent law. Using abstracts from the patent, it describes the invention as follows:

. . . a database search system that retrieves multimedia in a flexible, user-friendly system. The search system uses a multimedia database consisting of text, picture, audio, and animated data. That database is searched through multiple graphical and textual entry paths.

There is no intention, therefore, of limiting this invention to the exact abstract or disclosure presented herein. More particularly it is contemplated that this invention can be used by any information that can be stored in a database.

After that clarification, the area encompassed by this patent should definitely be vague to you if it wasn't already. I know it is to me.

Although I've read a portion of the actual patent, I'm not an attorney, so I can't tell you whether Compton's claims will hold up when subjected to the scrutiny of the legal beagles who will most certainly get fat representing the thousands of companies that are apparently affected by this little toll booth on the world's Digital Highway. But I will say that the portion of the patent I read looks like it describes almost any database retrieval mechanism available today. (Once again, I'm not an attorney, so don't take my word for it.)

So, depending on how broadly our technically knowledgeable legal system interprets the validity of Compton's claims, it is possible that almost anyone who is selling, or has ever sold, CD-ROM–based "multimedia" technology will be subject to one of the royalty structures offered by Compton's. Public relations contact Pat Meier was quick to point out to me from Comdex after this news hit the wires that the patent could cover anyone using almost any platform (including interactive television and personal digital assistants) to deal with databases of information.

I'm not sure what they do in the patent office for refreshment, but I have the feeling they'll be needing a bit of the "hair of the dog" to recover from whatever they were consuming when they issued this patent. Especially after the suits hit the fan.

Oh No, Speed Bumps Too!

I'm not sure how many of you are aware that San Jose's home town newspaper, the *San Jose Mercury News,* is available by modem through America Online, but it is. All you currently have to do to try it out free for a month is to call customer service at (408) 297-8495 from 7 A.M. to 7 P.M. weekdays (Pacific Time) and 8 A.M. to 2 P.M. weekends.

The idea of being able to get a daily newspaper on line is a good one for those of us who spend most of our time in front of our computer screens anyway. And for those of us who receive the daily paper by gas-guzzler net (the large vans driven by our newspaper delivery folks), having access to even more information than can be found on the printed page is totally cool (such information is indicated by little tags at the ends of stories in the hard-copy paper).

The idea of having newspapers available on line is apparently such a good idea that other papers, such as the *Washington Post,* hope to soon offer their readers on-line access to their pages through a publication called *Digital Ink* slated for delivery to a test readership of several hundred in mid-1994.

The main difference between the *Post*'s and *Mercury*'s current offerings, from what I've been able to discern, is that the *Post* wants to wait until it is able to graphically reproduce page content in a way that provides both low-cost text and graphics without compromising either.

There's the rub.

Current on-line technology does not lend itself well to the timely presentation of both text and graphics. The *Mercury News,* for example, uses place holders for graphics rather than attempting to provide them in an on-screen format—a wise decision when one considers the connect-time costs involved in providing graphics at 2400 baud, or even the 9600 baud offered at a premium by some on-line services.

The ability to present printed pages electronically for a reasonable cost to the consumer is a very large developer opportunity. According

to Newsbytes, an on-line news service, the *Post*'s *Digital Ink* electronic version alone is expected to attract about 150,000 subscribers.

The company that comes up with a method for quick, simultaneous reproduction of both text and graphics by means of standard on line connection mechanisms will make a pile of cash. Believe me, there are dozens of publications that would love to be able to provide their entire print product on screen without having to charge an arm and a leg.

For those of you who are wondering, Apple DocViewer, Adobe™ Acrobat, and other current electronic distribution methods, while good solutions for on-line delivery of many kinds of publications, are not able to accommodate the immediate needs of daily on-line newspaper delivery. The market really is wide open.

Whatever Happened to SLIM?

Many moons ago I was the evangelist for the original, ill-fated Apple Macintosh Portable. At that time (back in 1988), we were considering the inclusion of what today are called *Personal Computer Memory Card Industry Association* (PCMCIA) cards as part of the system's technology package.

Time has proved that we were ahead of ourselves in trying to include this technology back then, because it was expensive and its uses not completely understood. But one thing was certain: The name we had for those little credit-card-style memory cards was much better than today's PCMCIA. For want of a better term, we simply called them *SLIMs,* a name that described the look, feel, and use of the product.

Ordinarily I'm not one for turning back the clock, but in this case I'll make an exception. *PCMCIA* is absolutely one of the worst technology names this industry has ever come up with. I vote we either come up with a better name, or simply go back to calling them *SLIMs.* ♣

Amanda Hixson is currently a consultant in the area of product and process management. Along with being a five-year Apple alum, during which time she was, among other things, an evangelist, product marketing manager, and software project leader, she is also the author of four books and a successful CAI training tool, a journalist, industry analyst, former book acquisitions editor, accounting manager, and perpetual realist (or cynic, depending on whom you talk to).

Strategy Mosaic

QuickDraw GX

continued from page 1

"So what?" you ask. "I don't need all that fancy stuff. QuickDraw GX is for high-end applications, not me." Granted, some of the most spectacular things that will be done with QuickDraw GX will be in high-end publishing, drawing, and presentation programs. But regardless of what your program does, it will benefit from using QuickDraw GX. Why? Because type is what your user sees all the time, on the screen *and* on the printed page. This is true even with drawing programs, because when they use type, they usually demand more of it than most text-oriented programs.

And you have two more reasons for adopting QuickDraw GX now. First, if you don't, your product won't look as good as your competitor's. (For printing and graphics reasons why you should implement QuickDraw GX now, see last month's Strategy Mosaic.) Second, future Apple technologies will depend on the presence of QuickDraw GX; if your program doesn't use QuickDraw GX, you'll be behind the competition in an important way.

The rest of this Strategy Mosaic describes some of the major benefits of QuickDraw GX typography. (Even then, it'll be just the tip of the iceberg.) But first I have to cover a bit of background.

Characters and Glyphs

As QuickDraw GX sees it, characters and glyphs are two different things entirely. A *character* is the one- or two-byte code that represents an element of writing as the computer stores it internally. A *glyph,* on the other hand, represents a unit of writing as it is displayed visually on screen or paper.

A glyph can represent more than one character—for example, the glyph "æ" represents two characters, "a" followed immediately by "e". In some cases (as with the Arabic script), one character may have several alternate glyphs, with the correct glyph determined by the characters before and after it. Even in the Roman script (used by English, Spanish, and other western European languages), some fonts use alternate glyphs at the beginning or end of a word or line.

With pre-QuickDraw GX technology, it's too hard to keep track of anything but a one-to-one mapping of characters to glyphs. To do otherwise, your application would have to keep track of what font it's using, what "odd" glyphs it contains, and what character combinations are to be replaced by each special glyph. You'd also have to be constantly checking your user's input, searching for letter combinations and other contextual situations that would trigger the use of special glyphs. And what in the world would you do about the text-insertion cursor, if the user clicks in the middle of an "æ" glyph?

QuickDraw GX handles all this, and a lot more. In essence, *it automatically handles formatting and character-to-glyph mapping* so that your program doesn't have to. Your program "sees" a sequence of character codes that remains the same, regardless of what glyph combinations appear on the screen or page (see "Same Characters, Different Glyphs"). QuickDraw GX works with QuickDraw GX fonts (also called *TrueType 2.0 fonts*) and "does the right thing"

Same Characters, Different Glyphs. Depending on the font's glyph repertoire and feature set, the same string of characters can be displayed in different ways. Note the differences in the "N"s, "K"s, and "p"s. See the text for more details.

as influenced by the capabilities of that font.

Typestyles and Variation Axes

A *typestyle* is a variation in the appearance of all the glyphs in a font; common typestyles include bold, italic, thin, and condensed. QuickDraw GX fonts include the possibility of *variation axes,* variables whose values consistently change the appearance of a font. Apple defines four variation axes—*weight, width, slant,* and *optical size* (the last of which defines the optimal shape for a specific point size)—and allows developers to define and register their own variation axes.

Consider the example screen shot labeled "Skia Typestyles." The QuickDraw GX Skia font has two variation axes, weight and width. By varying them, you can create vastly different glyphs from the same font; this happens immediately, without the creation of a separate font. (Variation axes make fonts more useful, and your users will really like it if you let them change these axes from within your program.)

Ligatures

Ligatures occur when two or more glyphs combine to form a single glyph (for example, combining "a" and "e" to form the diphthong "æ"). Common ligatures include "ff," "fl," "ft," and "ll." You won't see ligatures in the main text of computer books and magazines, but they're commonly used in mainstream books.

Ligatures can be built into a QuickDraw GX font. If your program uses QuickDraw GX *layout shapes* to display its text, QuickDraw GX automatically substitutes whatever ligatures have been designed into the font. And guess what? You can configure your program so that when the user clicks in the middle of a ligature, the cursor appears between the two halves of the ligature and the ligature disappears if the user inserts a letter or backspaces. Also, since the underlying characters in the computer's memory haven't changed, text searches and spelling checkers still work correctly together—which *doesn't* happen today if you substitute ligature glyphs for the characters they represent.

Contextual Forms

As mentioned above, a single character can have multiple glyphs. The Arabic script, for example, has four separate glyphs for the character represented in English as "ha"—one glyph if it stands alone, and three others when it is at the beginning, middle, and end of a word. In the Roman script, font designers can make a font more elegant by providing glyphs that are used solely when a character occurs at the beginning or end of a word or, alternately, a line of text; these are called *initials* and *finials.*

To see the difference that contextual forms make, see the figure "Same Characters, Different Glyphs." Done using the QuickDraw GX Apple Chancery font, the three lines of text differ only in the font options that were turned on when the text was typed. The first line shows the effects of alternate glyphs at the beginning and end of the line. (Note the final "p" and the difference between the first and second "N.") The second line shows the effects of alternate glyphs at the beginnings and endings of words. (Note the "K" and the "p.") The Apple Chancery font has four *design levels;* the third line shows the effect of turning on both word and line finials and switching to the highest design level (level 4).

Kerning

Typography is an art, not a science, largely because of what the human eye (and brain) deems aesthetically correct. For example, in some fonts, the glyphs in the word "AWAY" seem to be too far apart, even though their spacing is mathematically correct. The process of manually adjusting the spacing between two glyphs to achieve a given effect is called is called *kerning;* together, the two glyphs are called a *kerning pair.* A font can store a table of kerning pairs, each entry of which says, in effect, "When glyph 'X' appears just before glyph 'Y', move 'Y' a certain amount closer to (or further away from) 'X' than it otherwise would have been."

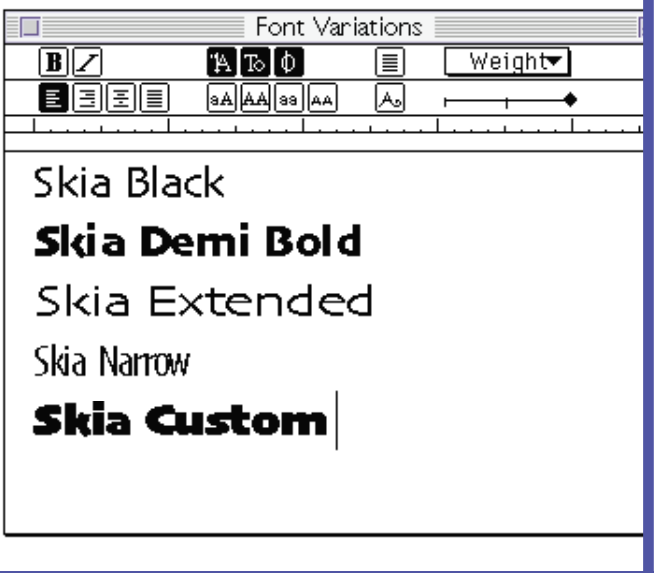
Kerning is another thing that QuickDraw GX takes care of automatically. If a QuickDraw GX font has kerning information in it (and all proportional fonts should), QuickDraw GX uses it to produce text that looks more natural than a font that does no kerning.

By setting a variable called the *track number,* your QuickDraw GX program can uniformly increase or decrease the space between all glyphs. This font-independent method of kerning is called *track kerning.* QuickDraw GX also allows *cross-stream kerning,* which allows individual glyphs to move perpendicular to the direction of text; this method of kerning can be useful for diagonal text and other graphic effects. And remember, regardless of the typestyle, ligatures, contextual forms, and kerning used to create the visible text output, the string of character codes inside your program remains the same. This means that your code remains simple, even though the text as displayed can be quite complex.

Other Features

QuickDraw GX has far too many features to list here—the current draft of *Inside Macintosh: QuickDraw GX Typography* is about an inch thick—but I'll list some of the more interesting things that QuickDraw GX makes possible:

• *Support for multiple, non-Roman scripts.* QuickDraw GX can handle virtually any script (writing system) in use today. It can handle right-to-left, left-to-right, and top-to-bottom scripts, multiple scripts on the same line,



Skia Typestyles. All the type samples in this GXWrite word-processing screen shot come from the same font, Skia. The first four are predefined typestyles, while I created the fifth one "on the fly" by varying the font's width and weight. (The slide control just below the word "Weight" allows you to change the selected axis of the Skia font.) GXWrite and lots of fun demos and QuickDraw GX fonts are available on the November 93 Developer CD.

even scripts in which adjacent characters sometimes change position.

• *Cursor and highlighting support.* In general, QuickDraw GX greatly reduces the work needed to determine where to place the cursor (hit testing), how to draw the cursor (vertical, slanted, or split), and how to highlight text. These issues become more difficult in situations where some text runs left-to-right and adjacent text runs right-to-left; in such a case, the cursor may need to be split, and a contiguous range of characters may appear as two discontinuous rectangles of glyphs. (See *Inside Macintosh: QuickDraw GX Typography,* "Carets, Highlighting, and Hit-Testing for Layout Shapes," for examples of this. You can find drafts of this and other QuickDraw GX documentation on the November 1993 Developer CD.)

• *Line break support.* Although the decision on where to break a line of text is still your program's responsibility, QuickDraw GX gives you routines that determine potential break points and the line lengths to those points; you can use this information to help make your decision.

• *Alignment and justification.* Alignment specifies the placement of the text in relation to the text margins; text can be either centered or left- or right-aligned. Justification deals with stretching or compressing a line to fit a certain width by adding space between words and sometimes between individual glyphs; text can be unjustified or partially or fully justified.

With QuickDraw GX, you set parameters that describe the alignment and justification you want, and QuickDraw GX takes care of drawing the text for you. The QuickDraw GX font design is the first to include provisions for more sophisticated justification. When a line of text needs to be "stretched out," QuickDraw GX adds spacing between words to a set limit; if it reaches the limit but needs to stretch the line of text more, it then starts adding space between characters. (This behavior can be varied and depends on certain information being present in the QuickDraw GX font being used.)

• *Numbers.* QuickDraw GX distinguishes between uppercase and lowercase numbers, as well as monospaced and proportional numbers (see "Number Styles").

• *Fractions.* QuickDraw GX and the proper font can work together to create fractions that use either a horizontal or diagonal fraction bar. Either approach looks a lot better than today's method of using uppercase numbers and the slash symbol (for example, "3/8").

• *Tangents.* If you use QuickDraw GX *glyph shapes* to draw text, each glyph can have its own position (either relative to the previous glyph or in absolute coordinates) and its own *tangent vector* (which specifies the rotation and size of that glyph). Using glyph shapes and tangents, you can quite easily implement text on a curved path and wavy text that varies in both size and direction.

• *Subscripts and superscripts.* QuickDraw GX makes it easier to manipulate subscripts, superscripts, and other glyphs offset from a line of text's normal baseline. It also allows you to manipulate such glyphs with greater precision than was possible before.

• *Resolution-independent text.* Since QuickDraw GX fonts use Apple's TrueType outline font technology, the text that QuickDraw GX draws will look as good as possible at any size, on any medium, and at any resolution.

"Smart" Fonts

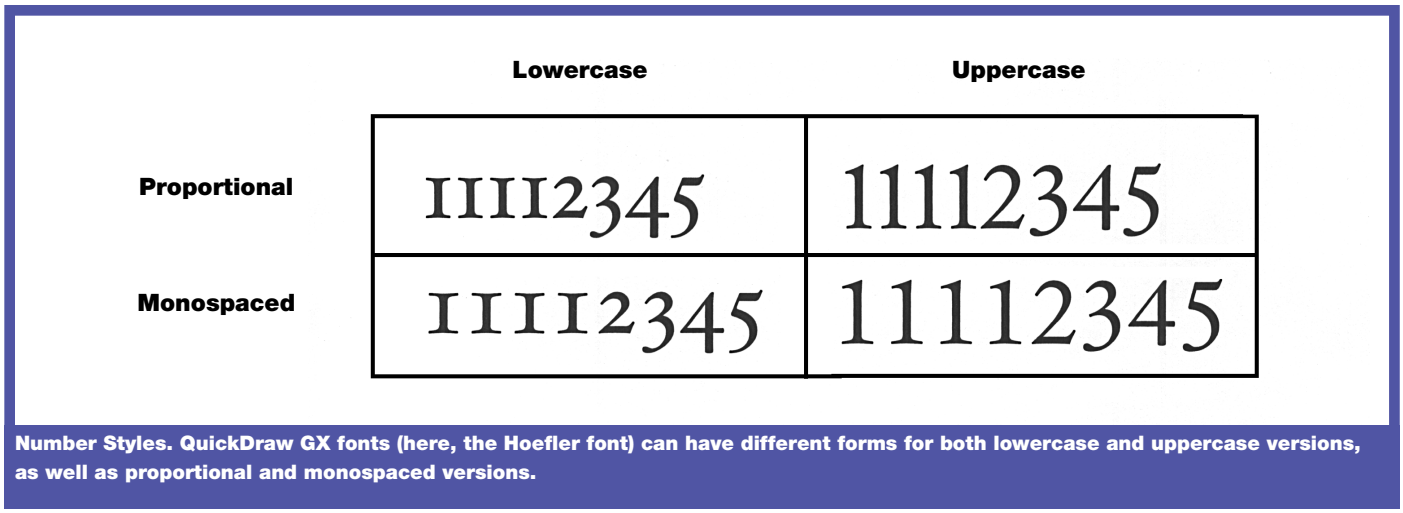
An important advantage of QuickDraw GX fonts is that, with so many kinds of typographic information built into the fonts themselves, people using them are more productive—the font takes care of details they used to do manually—and they make fewer mistakes. If you don't believe me, listen to Matthew Butterick, a font designer for The Font Bureau, Inc., a company that has designed fonts for *Newsweek, Time,* Apple, and Microsoft, among others. QuickDraw GX, he says, will allow his company "to roll much of the typographic intelligence of an art director into the font. And then production managers don't have to spend hours looking to see if someone used the correct set of figures or real small caps or other typographic niceties—the font does almost all the work."

In the October 1993 issue of *Publish* magazine, senior associate editor Gene Gable singled QuickDraw GX out as an important new technology. Like Butterick, he also pointed to the importance of QuickDraw GX's "smart" fonts. "A new generation of smart fonts that will make excellent typography more the norm than the exception should be hitting the market soon. In QuickDraw GX, Apple is finally supplying the tools that make it possible to embed creativity and expertise at the system level."

QuickDraw GX Type in Everyday Use

QuickDraw GX is much more than just type, but I can think of three good type-related reasons why you should use QuickDraw GX.

The first is simple: People want their documents to look good. Just by using QuickDraw GX layout shapes, your program gets



Number Styles. QuickDraw GX fonts (here, the Hoefler font) can have different forms for both lowercase and uppercase versions, as well as proportional and monospaced versions.

all the advantages of QuickDraw GX fonts, including ligatures, automatic kerning, contextual forms, proportional and monospaced numbers, "smart" quotes, and fractions—all of which transfer the font designer's expertise to every document your customer creates. Once people see good QuickDraw GX fonts, they'll want them, and they'll demand programs that use them.

But QuickDraw GX offers more if you explicitly add new GX-based features to your program. What user wouldn't want to create just the right font variation by directly adjusting a font's variation axes? Many QuickDraw GX fonts will have several design levels, and users will want to pick the one they like the most. (Compare the first and third lines of "Same Characters, Different Glyphs" for an example of the same font drawn at two different design levels.) And glyph shapes and tangents make text-on-a-curve and other typographic flourishes easy to implement.

QuickDraw GX does not force complexity on your users. You can write your program to provide whatever typographic controls are appropriate for your users. Complex, design-driven applications can provide new, highly exacting controls, while more mainstream products can hide the complexity of working with text and simultaneously provide better output automatically.

The second reason for using QuickDraw GX is related to both typography and printing. With QuickDraw GX, your program can create portable documents that any other Macintosh user with QuickDraw GX can view and print, even if he or she doesn't have the original application or fonts. Such documents are called *PDDs,* or *portable digital documents.* You can learn more about them in *Inside Macintosh: QuickDraw GX Printing.*

I should add that QuickDraw GX gives some document portability even to pre–QuickDraw GX applications. When running on a Macintosh that has QuickDraw GX installed, such applications create *spool files* that act much like PDDs. Users without the original application can view and print spool files, but they must have all the fonts used in the document's creation.

The third reason for using QuickDraw GX is to prepare your program for the future. Apple will, sometime in the future, add new technologies to the Macintosh that will require QuickDraw GX. If you adopt QuickDraw GX now, your program will be competitive tomorrow, and you'll be poised for future growth later.

High-End Uses

If you create fonts, font-manipulation utilities, or drawing, page-layout, word-processing, or presentation programs, you stand to benefit the most from QuickDraw GX. Your customers are insatiable—they want anything that allows them to do more with less effort, and they'll buy anything that makes their work stand out visually. QuickDraw GX gives you typographic tools that you, by yourself, could not afford to design, implement, debug, maintain, and enhance. And since QuickDraw GX is system-level software standardized by Apple, the products you create will, where it makes sense, be compatible with other developers' products. This will add to the overall richness of the QuickDraw GX environment and enlarge the market for your products.

It goes without saying that the best uses for QuickDraw GX type are yet to be invented. But here are some ideas to get you started:

- *Video and animated fonts.* This is a hot new feature I guarantee you'll see in tomorrow's programs. It works like this: First, you (or a font designer) create a QuickDraw GX font that has multiple glyphs for each character. Then you modify your presentation (or other) application to redraw text in that font several times a second, each time using a different alternate glyph for each character. The result is a presentation slide where the text on screen pulses, changes color, vibrates, or whatever you've designed the font to do.

  But why stop with characters? Create *animated fonts* in which the alternate glyphs of a "character" look like a person riding a bicycle, a sea gull in flight, a bouncing ball, or whatever else you can imagine. Sound far-fetched? Then look at the animated fonts included with the 1.0b2 version of QuickDraw GX that's on the November 1993 Developer CD.

- *Graphic manipulation of editable text.* Since QuickDraw GX text is a *QuickDraw GX shape,* it can be manipulated in the same ways as any other QuickDraw GX graphic element—including coloring, rotating, skewing, and other transformations. The resulting image is still QuickDraw GX text, which means that it can still be highlighted with a text cursor and edited. Believe me, this is a feature that graphic designers and artists will kill for.

- *Greater document reuse.* QuickDraw GX includes resolution-independent graphics as well as text. This means that if you use QuickDraw GX for all your application's imaging, your customers' documents will be more reusable than QuickDraw-based documents are today (in particular, bitmaps and, sometimes, PICT images are handled better under QuickDraw GX). Users will be able to take text and images from, say, a presentation program, then resize, rotate, and skew them as desired for reuse in a printed document.

Arriving Soon

QuickDraw GX will be commercially available in the first half of 1994, and you should start working with it now. In addition to QuickDraw GX 1.0b2 itself, the November 1993 Developer CD also includes Apple DocViewer versions of all the *Inside Macintosh: QuickDraw GX* documentation. Sure, it's a chunk of documentation, but QuickDraw GX does tons for you—and you didn't have to write the code, either.

QuickDraw GX will be the graphics and typography engine for the Macintosh through the end of this decade. Don't you think you should start using it? Your competitors will. ♣

Apple News

DOS-Compatible Quadra

continued from page 1

any product availability sometime in the first half of calendar year 1994.

For you, the new model indicates two important things. One, Apple plans to expand the overall Macintosh installed base—which means a bigger customer base for your Macintosh products. Two, if your company also sells DOS or Windows products, you will be able to sell all your products to this new market.

Audience

Apple believes that the Macintosh Quadra 610, DOS Compatible version computer will be of great interest to home office workers, small businesses, and other professionals. But Apple hopes this new computer will bring the Macintosh environment into new markets as well. One such market consists of current DOS users who need to upgrade their 80286 or earlier DOS computers to run Microsoft Windows.

Another market consists of DOS users who want the Macintosh ease of use but who own

DOS software. (Many DOS programs support vertical markets—scheduling and billing packages for dentists, for example—and, because of the small market size, will probably never be translated to either the Macintosh or Windows environments.)

One other market consists of first-time buyers who are apprehensive about buying either a Macintosh or a DOS/Windows system—Apple is rightly billing the Macintosh Quadra 610, DOS Compatible version computer as "the most compatible Macintosh yet."

Operation

Users of the Macintosh Quadra 610, DOS Compatible version computer can switch between the Macintosh and DOS/Windows environments with a single keystroke. (The Macintosh Quadra 610, DOS Compatible version computer will come with MS-DOS 6.2 preinstalled, and users can add Microsoft Windows 3.x, just as they would to any other DOS computer.) If the computer has two screens connected, the Macintosh environment occupies the startup screen, and the DOS environment occupies the other. (The PDS card has its own video-out and joystick port, so you don't need an extra video card to drive a second monitor.) If the computer has only one screen, it switches between the two environments.

Though only one environment can receive the current mouse and keyboard commands, both environments run simultaneously and independently of each other. Subject to the constraints of the programs in use, the computer supports the cutting and pasting of data between Macintosh, DOS, and Windows programs.

Implementation

The Macintosh Quadra 610, DOS Compatible version computer has a 25-MHz Motorola 68LC040 processor on the main logic board and a 25-MHz Intel 486 SX processor on its PDS card. It can use a specified portion of the Macintosh computer's main memory, or the user can install a 32 MB SIMM (single in-line memory module) on the PDS card for slightly better performance.

The Macintosh Quadra 610, DOS Compatible version computer implements an MS-DOS C: hard disk (and, optionally, a D: disk) as a disk file on the computer's hard disk. The Macintosh and DOS environments can open each other's disk volumes, thus making Macintosh/DOS file conversion as easy as dragging a file between two Macintosh windows. In particular, users can open the DOS hard disk in a Macintosh window; with the Macintosh PC Exchange software, users can open PC files from the Macintosh environment.

The computer works with the Apple 14-inch and 16-inch Macintosh Color Displays, as well as most VGA, SVGA, and multisync monitors. The CNT BIOS (basic input/output system) implements DOS compatibility, and Apple is working closely with Microsoft to ensure compatibility with Windows and with Microsoft applications.

Apple may make variations of this technology available, depending on customer and developer feedback. The version shown at Comdex is targeted toward people working by themselves, usually in a business context. For this reason, it does not include features such as networking on the IBM side (it can network through AppleTalk, on the Macintosh side) and support for sound-enhancement cards (like Sound Blaster).

Apple Announces AppleScript Scripter's Kit, HyperCard 2.2

On December 13, Apple announced a commercially available version of AppleScript and a new version of HyperCard® that supports scripting and the creation of stand-alone applications from HyperCard stacks.

The AppleScript Scripter's Kit, with a suggested retail price of $189, is meant for business and individual "solution developers" who intend to do extensive application scripting. It lacks the CD-ROM, development tools, Scriptable/Recordable Finder, sample code, and electronic documentation found in the AppleScript Software Development Toolkit version 1.1.

The AppleScript Scripter's Kit also includes Frontmost, an interface builder from Software Designs Unlimited, Inc. Frontmost allows you to create a "front-end" application for the scripts you've built. A stand-alone Frontmost application has an overhead of about 210K.

HyperCard 2.2, to be distributed by Apple at a suggested retail price of $249, includes several major improvements. By supporting the Open Scripting Architecture (OSA), HyperCard 2.2 allows you to write HyperCard scripts using either HyperTalk® or any OSA scripting system—Apple's AppleScript or Userland's Frontier, for example. In this way, your stacks can use the services of Apple event–savvy programs like Microsoft Excel 4.0 and Filemaker® Pro 2.0. HyperCard 2.2 itself responds to a variety of Apple events, so you can now control it from any application that can send Apple events.

If you are using System 7 (HyperCard 2.2 runs with System 6.0.5 or later), you can save a HyperCard 2.2 stack as a stand-alone, royalty-free application that does not require the presence of HyperCard to run. (Such an application also lacks the HyperCard script editor; it runs as its stack equivalent would, but the user cannot access or change its scripts.) This feature makes it easier and more convenient for users to distribute solutions created using HyperCard.

HyperCard 2.2 is compatible with System 7.1 WorldScript. This means that HyperCard 2.2 can display text in scripts (writing systems) such as Japanese, Arabic, Korean, and Traditional Chinese. When the appropriate script is installed on the computer, HyperCard 2.2 can handle script features like double-byte characters, bidirectional text, and contextual ligatures.

HyperCard 2.2 will ship with a stack that allows you to color HyperCard buttons, fields, backgrounds, and cards, use color images as buttons, so color image-transistor effects, and make other use of color in stacks.

HyperCard 2.2 contains numerous other improvements, including interface elements that look like standard Macintosh interface elements, built-in support for button clusters (called *button families*) and pop-up menus (called *popup buttons*), new report-printing features, Balloon Help for HyperCard 2.2 itself, four new visual card-transition effects, and undo capabilities for button and field deletions.

Scriptable Finder Available

Developers wanting to use AppleScript to coordinate the work of multiple applications have been

hampered by the current Finder's inability to be directed by AppleScript (or any other program that uses Apple events). A scriptable Finder will not be available to customers until sometime in the first half of calendar year 1994. However, you can now get the same Scriptable/Recordable Finder as will ship later in 1994 so that you can work on products that make heavy use of AppleScript.

Apple is including this new version of the Finder (and documentation for its scripting features) in the AppleScript Software Development Toolkit version 1.1, available from APDA (R0175Z/B, $199 in the U.S.). This new version of the AppleScript Software Development Toolkit, which should be available by January 1994, includes four floppy disks, a CD-ROM, three printed manuals, and a redistribution license for the AppleScript system software. This product includes the AppleScript extension software, sample code, debugging tools, script editor, and electronic documentation. Owners of the previous version of this product can order the AppleScript Software Development Toolkit version 1.0 to 1.1 upgrade (R0557Z/A, $99 in the U.S.) See page 28 for information on contacting APDA.

PowerPC Update: System Goes Beta

If you're like a lot of developers who've written us at *Apple Directions,* you're excited about the market and performance potential for products that take advantage of the forthcoming PowerPC processor–based Macintosh computers, but you're not quite sure how to get started with them.

At next month's Macworld in San Francisco, to be held January 5–8 at Moscone Center, Apple Computer, Inc., will be able to help you get started developing for Macintosh with PowerPC, if you haven't started already. You'll want to come by Apple Developer Central at the show to find out more and get answers to your questions about Macintosh with PowerPC. (See "Visit Macworld Developer Central" on page 11 for more information.)

In the meantime, Apple has made several announcements about the new platform:

- System 7 on PowerPC has reached the beta stage of development and is now undergoing final testing.
- Six more key developers from around the world announced support for the Macintosh with PowerPC platform.
- An agreement between Apple and DayStar will result in an upgrade card that provides PowerPC technology to Macintosh Quadra 700, 900, and 950 customers.

"Reaching beta for System 7 on PowerPC means we're right on track for the delivery of the first Macintosh systems based on PowerPC in the first half of 1994," said David Nagel, general manager and vice president of the AppleSoft Division. "Things are really looking good."

Macintosh systems based on PowerPC technology will be compatible with thousands of current Macintosh software applications, which means that customers will be able to run existing Macintosh applications on newer systems based on PowerPC technology.

Applications written especially for System 7 on PowerPC (so-called native applications) will feature greatly enhanced performance.

"It took us less than a week to have an early version of Painter up and running native on a PowerPC processor–based Macintosh, and we have already seen speed improvements of two to four times versus the Intel version of Painter running on a Pentium," said Mark Zimmer, president of Fractal Design Corporation, developer of the Painter graphics application.

Fractal is among the growing group of developers from around the world who have committed to developing native applications for PowerPC processor–based Macintosh computers. Recently, six more developers added their names to the list, bringing it to a total of 24: Abvent SA, Brossco Oy, CTM Development SA, Dantz Development Corporation, Graphsoft, Inc., and Hi Resolution Ltd.

Another developer, DayStar Digital, has agreed with Apple to provide a new PowerPC 601 processor upgrade card for Macintosh Quadra 650, 700, 800, 900, and 950 and Macintosh Centris 650 computers. Under the agreement, Apple licensed its core system-level ROM code to DayStar so that they could develop the card.

DayStar said it plans to market cards that will operate at 66 MHz or 80 MHz and provide optional on-board memory up to 128 MB. DayStar's PowerPC processor upgrade is expected to be available at the introduction of the first Macintosh with PowerPC computers and Apple's PowerPC upgrade solutions in the first half of 1994.

Apple previously announced plans to offer upgrades for 11 Macintosh computers and for all three models of the Apple Workgroup Server products. The models included are the Macintosh Centris 610, 650, and 660AV computers; the Macintosh IIvx and IIvi, Performa 600, Quadra 610, 650, 660AV, 800, and 840AV computers; and the Apple Workgroup Server 60, 80, and 95. Apple plans to make upgrades available with prices starting below $1,000. Specifics on various upgrade options are expected to be announced at the introduction of the new Macintosh with PowerPC technology– based systems.

We'll have more to say about how you can join the Macintosh with PowerPC bandwagon in next month's *Apple Directions.* Until then, if you're going to San Francisco, be sure to visit Apple Developer Central at Macworld for the latest PowerPC information.

New Services to Help You Find Customers

Among the toughest challenges facing computer-related service and solution providers is finding and keeping clients for their services. The market for these services is huge, but fragmented and expensive to reach through traditional advertising channels. How can small to medium-sized companies gain exposure and business without spending a small (or large) fortune?

The MZ Group,Inc., a publishing and new media company based in San Francisco, is helping to answer that question by building a new solutions channel that may fundamentally alter your entire sales and marketing effort. Having published the popular *Macintosh Services Directory* since 1991, they are now expanding the directory to a multiplatform format and adding a "matchmaking on demand" referral service to better serve the tens of thousands of businesses that use the book as a resource. These changes offer unique marketing opportunities and service resources for Apple developers, service providers, and customers.

Renamed *Agora* after the marketplace of ancient Athens, the directory has been expanded by MZ Group into a "marketplace" offering a number of services that connect buyers with providers of services and solutions. This allows for an increased circulation and an economical way to match qualified service and solution

providers with those who have their services.

Launched in 1991 in cooperation with the Apple Developer Group, the *Services Directory* was the first publication exclusively devoted to service providers. About 700 companies were listed in the first directory; since then it has grown to 2300 companies in 150 categories ranging from C++ programmers to multimedia animators.

Responding to changes in the computer industry, MZ Group has retooled the directory. "We talked with users of the directory and drew valuable intelligence that guided our changes," said Gary Parsons, director of marketing at MZ Group. He listed them:

- Solution buyers prefer a single guide rather than a number of smaller ones because most projects require expertise and skills in different areas and on different platforms.
- Solution buyers want help in identifying providers, since they aren't always sure about their alternatives and which providers to call.
- Providers want a steady stream of buyers for their services and need to reduce the peaks and valleys in their business to avoid being either too busy or without work.

MZ Group's Agora product offers two main components that make it easier for you to build awareness and find customers.

The *Agora Services Guide*

The *Agora Services Guide* is a multiplatform resource guide used by information system executives, systems professionals, managers, and developers looking for solutions and service providers.

Published twice a year, it lists thousands of companies in over 150 categories. Major sections include Custom Solutions, Development and Programming, Marketing and Distribution, Multimedia, Networking, Support Services and Writing, and Production and Publication Services. The *Services Guide* is distributed worldwide and has a readership of over 100,000.

Providers of computer-related services and solutions can receive a free listing in the *Agora Services Guide* and information on increasing their exposure within the directory by contacting MZ Group at (415) 543-4600. Companies or individuals interested in locating qualified providers will be able to find the *Agora Services Guide* in over 5000 bookstores throughout North America, or they can call MZ Group to order a copy. The directory will no longer be included in the Apple Developer Mailing.

The Agora Referral Network

The Referral Network provides an easy way to find qualified service and solution providers. Buyers and providers can call (800) 927-1200 to speak with a referral counselor; this counselor gives buyers profiles of service providers with the precise skills and experience they are seeking, and gives service and solution providers referrals of qualified buyers interested in their services.

To receive these referrals, providers must become members of the Agora Referral Network. Members specify the types of projects they're looking for, and are then given the opportunity to sell their services to buyers looking for their particular expertise.

Companies or individuals who need services and solutions use the Agora Referral Network to find qualified, prescreened providers, without having to pay the hefty fees required by recruiters or agencies.

For more information on how to use Agora to get access to new clients, or to find qualified service or solution providers for your next project, call (415) 543-4600, fax (415) 543-8232, or send an AppleLink message to AGORA.

Additional Products Announced With System 7 Pro

The November issue of *Apple Directions* listed developers' products announced on October 4, the same day Apple Computer, Inc., shipped the new version of its operating system, System 7 Pro, which includes PowerTalk, AppleScript, and QuickTime 1.6.1.

Two developers' products were left off the list, an oversight we'd like to correct by mentioning them here.

GBI, Inc., released two products for System 7 Pro: AOCE Connection Kit for HyperCard and AOCE Connection Kit for 4th Dimension.

In addition, the Milum Software Group announced Office Tracker Pro, a group scheduling, task management, and in/out tracking product, and Things To Do, a multi-user to-do list application, both of which support PowerTalk and AppleScript. ♣

Technology

Inside This Section

Visit Macworld Developer Central for Info on New Apple Technologies

Macintosh with PowerPC, Newton, OpenDoc, AppleScript 1.1, HyperCard 2.2, PowerTalk, QuickDraw GX—that is, all the latest technologies from Apple Computer, Inc.—will be featured at Apple Developer Central, part of Macworld San Francisco to be held January 5–9.

At Developer Central, you can attend technical sessions about all the technologies, see demonstrations of development tools, and meet with Apple representatives to have your questions answered about Macintosh and Newton development.

Many of you have had questions about development tools for the PowerPC processor–based Macintosh computers, which Apple will begin shipping in the first half of 1994. Developer Central will be the place to go to have your questions answered and to learn how you can get started developing native products for Macintosh with PowerPC.

You'll also want to take advantage of special offers on the latest development tools, including the Newton Toolkit and new versions of AppleScript and HyperCard. A variety of other development products, including Developer University self-paced training classes, will also be available at Developer Central.

So stop by and get the scoop about Apple's newest technologies. You can find Developer Central at the mezzanine level of San Francisco's Moscone Center. ♣

CD Highlights

System Software Edition, January 1994

Hello and welcome to the January System Software Edition of the Developer CD, featuring 150 MB of new and revised system software, system enablers, technical documentation, tools, and utilities. Featured this month is a late beta version of an update to the Developer CD Contents Catalog that adds some of the capabilities requested in recent user feedback. Please copy it to your local hard drive and give it a try, and let us know what you think, either through this month's survey (in the CD Info folder) or through an AppleLink message to DEV.CD.

System Software Edition

The Developer CD Contents Catalog is a simple-to-use tool for searching and navigating the Developer CD Series. It consists of a sequence of information screens that provide the following information about each package on the CD:

- title
- brief description
- list of files in the package
- revision information

The Open Folder button on each package information screen allows you to directly open the package's folder on the Developer CD. The Browser button takes you to the Browser screen, where you can easily view and navigate through different sets of entries (for example, New and Revised packages). The Find button and menu command allow you to do full-text searches of this catalog's contents.

To get information about a package on the CD, follow these steps:

- Click the Browser button to go to the Browser screen.
- From the Current Selection pop-up menu, choose a group of package titles to display. As a default, new and revised items are displayed; you may also choose to display all packages, the results of the last text search, or the packages on a particular CD.
- Double-click an item in the packages list to go to the information screen for that package.
- Click the right arrow button to advance to the next package in the current selection, or the left arrow button to go to the previous package.

Here are some of the other new and revised packages on this month's CD.

FMAT Editor

This ResEdit editor lets you prepare localized number format descriptions that you can use in conjunction with the number formatting routines provided by the Macintosh Text Utilities. Using these number format descriptions doesn't give you all the flexibility of the approach described in the

Please turn to page 20

Human Interface

All I Want for Christmas

By Pete Bickford

I've been planning to write a holiday column for some time—a year, to be precise. Last Monday, I knew that the time had arrived: There was a slight nip in the Cupertino air, Comp-USA had put out their Christmas decorations . . . and my editor called, asking how my article was coming along.

Like I said—the time had obviously arrived.

In this column, instead of trying to tackle the great topics of interface (which in the past year have ranged from enterprise computing to comic books), I'm going to be selfish. I'm going to write about what I want for Christmas. Of course, none of the items on the list are for me, really. No, these holiday wishes are more of the "end to hunger" or "world peace" variety.

Stocking Stuffer Wishes

• *That all applications would arrange their OK and Cancel buttons properly.*

For the record, the button whose meaning is most like OK goes on the right—regardless of whether or not it's the default button. This allows users to anticipate which button to click and begin moving the mouse into position before the dialog box actually appears.

Reversing the placement of the OK and Cancel buttons wasn't such a big deal until the dawn of System 7. At that point, Apple decided to follow its own dialog-box placement guidelines and reworked literally hundreds of dialog boxes in its system software so that they all had the button order right.

Unfortunately, this sudden consistency makes any application that reverses the positioning seem intolerable. Right now, maybe 90 percent of all software gets the order right. I'll offer a free copy of ResEdit to any of the remaining 10 percent who want to take a few minutes and fix this simple, but annoying, problem.

• *That computer programmers would learn to use real quotation marks (“ ”) instead of inches marks (").*

Using inches marks (") was a malignant outgrowth of the limited number of keys on early typewriters. Of course, back then, it was also customary to use a lowercase *L* as a substitute for the number *1*. Such nostalgic practices have no place in the modern world of computing.

Folks who develop for those other computers may be able to get away with these sorts of typographical misdemeanors, but Macintosh folks are supposed to be hip to things like fonts, proportional spacing, and typographical marks. Columnist John Dvorak said that before he even reads it, he can tell that a letter came from a Macintosh user instead of an IBM-PC user. I think we can all guess why: The Macintosh owner's letter looks better.

The Big-Ticket Wishes

• *That new software releases would concentrate as much on getting old features right as they would on adding new ones.*

A year ago, I ridiculed a certain word-processing application for ballooning to 17 MB of disk space. I recently learned that the next release of this program is expected to exceed 28 MB on disk. Having run through numerous major versions, this word-processing application has added everything from built-in charting to a (really terrible) drawing program. It's almost as if the developers thought that they had to have an entry on each line of some imaginary feature checklist. That this barrage of poorly implemented features obscures the fundamental function of the program—writing—seems to have been lost on this program's developers.

Unfortunately, this is far from the only program to practice gigantism instead of perfection. Developers of one of the leading database programs for the Macintosh computer have spent recent years coming out with add-on word-processing modules (ironic, eh?), drawing programs, and even a three-dimensional CAD library. All the while, they seem unable to produce a standard pop-up menu or properly shaped push button to save their lives.

Come to think of it, these are two things that another program I'm fond of picking on—HyperCard—finally is getting right in its next release. I guess there's hope for the world after all.

• *That applications have fun from time to time.*

As the standards for software get higher, we have to remember to keep our sense of fun—particularly when we design "serious" applications. This doesn't mean that your software needs to crack jokes or anything, just that it doesn't need to handle every job with somber economy.

A good case in point is the Apple Installer. It's a practical program that does its utilitarian job using large-point type, friendly "counting hand" cursors, and animations of shuffling disks. Far from being criticized as "not being serious enough," the Installer gets accolades for making a sometimes frightening process seem nonthreatening—even downright amiable.

I'd also like to see software get a lot less formal when it comes to the prompts and dialog boxes they give to users. Some messages I've seen are so stiff, so pedantic that you'd swear they were written by civil servants. Lighten up a bit on the "The system state is such that the item you have selected . . ." messages, willya? And while you're at it, a few contractions wouldn't hurt (as long as you remember to use curved apostrophes!)

And now, the biggest wish of all. . .

• *That development teams would remember who they're designing for.*

If you're on a development team, chances are that someone on your project possesses a very sharp whip. That person's job is to

crack that whip with sufficient frequency that your product is herded out the front door precisely on schedule. In addition, your project team probably has people in charge of engineering, marketing, production, packaging, distribution, and so on. Sadly, these people might all be very talented, work very hard, and deliver their services in strict accordance to the cracking of the whip—yet you'll still have unhappy customers if you haven't designed a product that meets your customer's needs.

Each year, hundreds of products are created, and thousands upon thousands of "features" are implemented, that are fundamentally useless. Either they are of no interest to anyone except their programmers, or they're implemented in a way that's totally alien to the people who are meant to be using them.

Why does this terrible waste continue? In large part because potential customers are still held at arms length; often the only time they are asked for their opinion is when the product is ready to ship. The really bad (and generally short-lived) companies never bother to ask at all. (Joy Mountford has a lot to say about product design teams and the importance of listening to customers in her interview on this page.)

This holiday season, let's take some time and remember the true meaning of product development. It's not about pretty packaging or powerful code bases—it's about people. One hopes that it's about doing something that makes someone's life a little better, a little more productive, or—yes—a little more fun.

So there you have it: my own personal list of holiday wishes. Anyone care to start making New Year's resolutions?

Happy Holidays,
Doc
AppleLink: THE.DOKTOR

Pete Bickford is a member of the Apple Business Systems human interface team.

The State of the Macintosh User Experience, Part Two

The Realities of New Product Design

Joy Mountford has been an influential member of the Apple Computer, Inc., user experience community for seven years. She started her tenure in the system software management product group. After a year and a half, she moved to the Advanced Technology Group, where she managed the ATG Interface Group for five years. She's now leading the ATG Design Center.

Talking with Joy, you quickly realize two things: First, she really knows her stuff. And second, if you don't pay attention, you'll miss a key point or even an entire line of thought. Though we talked about many things, some of her best comments answered the very practical question "How do you go about designing a new product?"

After over five years of continually exploring that question, Joy is enjoying a change of pace: She is currently on a sabbatical that was two years overdue.

Apple Directions (AD): *Let's start off with the question "What are your people thinking of when they say 'user-centered'?" What does that mean to you?*

Joy Mountford (JM): I think it's a point of view and I think a lot of people have it and even more people don't have it. It's really more that, when you approach something you ask, "What's the goal of this?" and "Is the person who's using it actually getting what they need?" You have to keep asking that question all of the time, as opposed to a technology-driven way of looking at things, which is "This is sexy and it's great and people will love it and we'll figure out what they're going to do with it later."

We try to put it the opposite way and say, "Why do you want it? What's it going to do? Who's going to use it? Do we know that they're going to use it? How do we find that out?" The common answer to this is, "We go and ask them." Will they tell us? Probably not. They'll say they don't want anything, because they're happy today.

AD: *So how do you find out?*

JM: You have show them something and see—really see—how they react to it. There's an infamous story that, probably, a lot of people now think is humorous. When we first did the QuickTime controller, the people directing the project were from film school. And if you know much about film, you know that they edit their film by using a jog shuttle controller. So they did an analogous thing in software, which was what was called the *hand controller.* It was a visual hand that went along the control bar that you could use to grab and release the film, using the mouse.

The two guys that did it, loved it. But everyone else who was going to edit video, who wasn't from a film background, said, "Well, I'm not really sure I want that. You know, what's the hand doing?" OK. So they wouldn't believe this. Of course, people don't believe each other, which is good. So we went out and we did lots of studies. And, lo and behold, we collected all this data showing that basically everyone said, "What's the hand doing? I don't want it! I just want to use, basically, a linear control, in and out."

But when we showed this to the design guys, they said, "Well, we designed the hand controller

wrong. What we really should do is, we need to add sounds and we need to add, you know, a bit different animation, whatever." And so the psychologist [on our team] had to say to them, "Yes, but I think the point of this test is that people don't want the hand."

The point of this story is that, even in an interface community, where people are very sensitive to interface issues, they have their own blind spots and they won't listen to the user tapes well. Therefore, you also have to have objective people that come in there and say, "Look, we love it, but listen." The hand controller, as you know, is no longer there. And it took us about a year to get rid of it.

The Design Team

AD: *That story touches on something that Don [Norman] was saying, that there are two parts to user testing. There's the testing, but that's almost less important than the ability to hear what the test results are.*

JM: It's impossible for people to listen, so you can't have the same guy testing it who designed it. That's why you need different departments working on the project from the start. But the most important thing is that the group has to report to one common person who has that perspective. It doesn't matter who that person is, particularly. But if they report to two different people, all you get is, "Thank you for your information. Next?" because they don't have to listen.

AD: *If you were able to build the design team from the ground up, what would it look like?*

JM: Well, the way that we did our work in the Advanced Technology Human Interface Group was to have an interdisciplinary team. You can't have just software people or, let's say, just interface people. One person has to be, quite clearly, in the visionary role, analogous to what Steve Jobs was in the old days. Sometimes, it doesn't matter what the decision is, but there have to be very clear decision points, and someone has to be driving a vision. This person has to have a lot of common sense and the ability to work with a weird combination of team members. Within that team, I think you need to have tremendous respect between the various people on the team—the designer, the user psychology person, the testing person, the user studies person, and the programmers and industrial designers.

"You need different departments working on the project from the start."

AD: *Why industrial design?*

JM: Because nowadays, we've moved away from the desktop stuff. With Newton and things like that, you have a whole different meaning assigned to your computer. Is it something that you sit and work with? Is it something you prop up? So you have to have people who can also now build hardware prototypes. With a CD player design we did, for example, we handed some users a foam core about the right shape, gave them a bunch of buttons, and asked them to lay out where the buttons might be. This is where you really begin the design.

So what are the engineers doing meanwhile? Well, they're understanding the real problem for the first time. Let's say they've designed this fabulous CD player, really sexy. You take it out to a field mechanic whose hands are grubby—he can't use this thing. A pink case isn't going to work. The player buttons are, you know, miniature, and these guys have big fingers. So the engineers suddenly think, "Oh, this is in a context in which it will be used." And then they back off, and they start saying, "It doesn't really matter that we have the world's best touch screen. What really matters is that we have big buttons." Or, you know, whatever the tradeoff might be.

AD: *Are there times when the marketing side can cause problems?*

JM: Well, I think marketing has not been connected with engineering as well as it should be for new projects. Often, they only put a marketing guy on the product after it's already designed. I think we should be using marketing people to gather data about real people and say, "Where is there a niche for this product?"

Sometimes they do get that data, but they need to communicate that data successfully to engineers. Do they write a report? That doesn't really work, so they have to work with the engineers in person. They might even need to have pictures or video or something like that to explain to engineers, "This is what we found out, and here's the reason we're going after this." I think the problem with marketing has often been that it's perceived as selling, and sometimes engineers pull away from that.

The other problem is that marketing data typically is based upon what you find out about today. This is a problem in interface, too. What you're really trying to find out is, "What are things going to be like tomorrow?"

AD: *And next year.*

JM: Right. I think engineers should be influencing marketing people more by saying, "We think we could do these types of things. You know, it's possible that this could happen. Do you think people want to have that?"

AD: *This may be an impossible question to answer, but how do you balance the conflict between good design and quick time to market?*

JM: Well, yes, if we knew how to do that, we'd all be happy, right?

I think that you have to do some degree of quick-and-dirty stuff. People who have a lot of academic training find that tradeoff particularly hard, because you feel like you're doing a shoddy job. However, with experience, you notice that an awful lot can be changed, sometimes, with some smaller amount of effort than you might initially think.

So what you have to do is say to the designers, "You've got two weeks to make a prototype. Whatever you've got done in two weeks is great. Then we take it to the customer." And then, of course, the customer'll say it's terrible, and they're right. Great! So now let's begin a conversation with the customer. If you wait until the very end, you can't have that conversation. So that will help you get design earlier, better, faster, but it means that certain types of people have to give up parts of their egos.

Quick-and-Dirty Prototyping

AD: *When a company starts on a new product and wants to do this quick-and-dirty prototyping that you suggest, what sorts of tools and techniques do you find useful?*

JM: It's interesting, because I run the International Interface Design Competition with universities around the world, and one of the things we do is to get students from different departments to start working together. So every year, the engineering students say, "Well, we can't compete with the visual people." And the visual people say, "Oh, we can't possibly compete with the engineers." And they have this big debate about industrial design, because the engineering students often don't have access to milling machines to make fancy plastics. So a team that doesn't have access to something complains that they don't have a chance against one that does.

The people who won last year were engineering students—in fact, in a landscape architecture department—who built their mockups out of pizza boxes. So what they did was to say, "What can I get my hands on that makes sense?" They needed a box that was a certain size, so they decided to look around for something that was the right shape. They found that a pizza box worked for them—it was stiff enough, and they put fishing-tackle weights in it. They got the feedback they needed: When they gave it to people to hold, people told them immediately that it was too long, that it hit them when they bent their arms to carry it, and so on.

To me, that was a brilliant idea. There was no tool involved and they didn't need to mill anything, because the question they were asking was "Is this comfortable to hold?" So 90 percent of what people should be using is what I call the old "brown-paper-and-string" approach. You can make immense breakthroughs if you actually ask the right question and use whatever's around you.

AD: *What about prototyping software?*

JM: In terms of interface, that's a little different, because then you've got stuff on the screen. I have helped teach an interface class with Post-It notes and acting, because we can't have a bias toward the Mac or the IBM machine. We get people to "become" an interface, as it were. They move the notes around on the screen, take them off, put them back—they "act out" being the interface. So you can do prototyping even without computers.

Of course the wonders of things like Macromedia Director made our lives "better"—except for the problem that when you show something in Director, a lot of people think it's finished. The problem is to find a visual language that maps onto the maturity or "finishedness" of a project; this is a very difficult prototyping problem. So now we use Director, but we do what we call "drawing rough"—the artwork is sketchy, it's not straight-line.

AD: *To set expectations.*

JM: Right, and the really important thing about why that's been a brilliant breakthrough—in fact, one that we've written research papers about—is that people give you different types of feedback. Because if I give you this wonderfully polished tape recorder and I say to you, "Can you tell me what you think is wrong?" or whatever, you will make very small suggestions of change. You'll say, "Move that control over here," or "Make the case green." Now, if I give you a really rough prototype, you immediately know that I haven't finished designing it, and then you give me lots more information.

AD: *And another benefit is that the more sketchy a prototype is, the faster it is to make.*

JM: Correct.

AD: *You could spend weeks or months making something that looks totally realistic in Director—*

JM: Right—and then you find out that it's wrong, and you get mad and you don't want to change your ideas, because you've spent two weeks scripting it.

"I think sound is the big element that we're really quite ignoring."

The bottom line is that you should begin the simplest, cheapest, easiest, fastest way you can, and I argue that there's no point in discussing an interface more than for two hours. Once you've discussed it for two hours, I want to see it. You show it to someone, and they say, "That's not what I had in mind." And I go, "Great. Now, then, tell me what you had in mind." If we have this between us, it helps. But if you and I just keep using words, what I imagine and what you imagine are so completely different, it's impossible to use words to do it.

Changes in Work Styles

AD: *In terms of noncomputer electronic devices like personal digital assistants, do you see people using these devices differently? What can you say about people's work patterns?*

JM: I think that out-of-the-office, field-situation use is really an important one. Typically, we design for an office situation or for meetings. But I think what people are seeing is that, when you're on the road, you're in a range of situations. You've got time in different places and you have to be able to use your time well.

You want to take photographs of something, make a sound recording, things like that, and not be bogged down with power units and cables and so on. And then you want to gather that information together and use it for something.

If you look at what people do from the beginning to the end of the day, you'll find they want to have pieces that fit together. When I go on holiday, I take three cameras with me and pick which camera I want for each day. So people want to be able to mix and match, and I think that's where software needs to be. I think that the component-based model that we're all going toward incorporates that view.

There's this view that we sit down and create new content all day long. In fact, most of what we do is try and find things. We do lots of bits and pieces, and then we tweak it and add to it from other sources. It's iterative, and I think that this is one software market that really needs to grow answers to the question, "How do we gather stuff in a relatively loose form?" And then, over time, we start to mix and match bits and begin to produce something.

And I think the other thing that's very obvious is that people don't think of data as different types of data. They don't say, "I'm in text mode" or "I'm in voice mode," or "I'm in camera mode." We have to make things so that data types are concurrently available, because you and I see, hear, and think all at the same time. And tools are not really like that at the moment.

Missed Opportunities With Sound

AD: *Are there data types we could be doing more with?*

JM: I think sound is the big element that we're really quite ignoring. I think we're not doing enough with sound. One very interesting thing is that when we talk about "multimedia," most people talk about digital video in a window. They don't talk about sound. Imagine that you have some video on disk; well, it's got sound with it, too. And you probably want to scan through the sound, because what we most often want to find is the piece where someone said "X." How do we do that? We have no idea. Impossible. How do we ask and search and remember, and store and find sound?

We actually designed a tape recorder that let you overlay tones—on one track what we called *audio stamps*—when something interesting happened. Once we had a tape like that, we digitized it into the computer and used software that let us fast-forward the tape and play it at twice its normal speed without getting that chipmunk effect. Then the user could go directly to the markers.

Why don't developers just build some of this stuff? We've built it. We can't get Apple to ship it, because people don't think sound is important. So the developers have an open market, as far as I can see, in that whole arena. Some of the architecture's there. The Sound Manager's there. [Editor's note: See "Sound: The Final Frontier" on page 17 of this issue for details about the latest version of the Sound Manager.]

If I had a tape recorder of the type that we built, I would use sound more. I record meetings all the time, but I can't find anything, and I haven't got hours to go through the whole tape.

If multimedia is going to stay, sound has to be there. When film was invented, you had moving pictures and cameramen, but there was no sound attached to it. Remember that? It's the same as what we have today, with computers. When sound came along, the sound guys earned more money than the film guys, because so few people understood sound.

Here's a good piece of trivia. You show people HDTV [high-definition television] with mediocre sound, and then you show them really good stereo sound with an OK picture. Ask them, "Which has the better picture?" And what do they answer? The one with the best sound, not HDTV. Why is this data screaming at us, and no one's doing anything about it?

AD: *Sound can convey a lot of things that can't be said verbally. And that's another thing we don't understand—what does the sound mean? We could use sounds as reminders, as prompts . . .*

JM: There's a lot we did with using background sounds to convey information—for example, the Sonic Finder. The problem is, background sounds are usually relatively personal. When you demo them, people turn the volume up, and then they say, "Well, that's really stupid." And you say, "Yes, but imagine it low-key." If you do it low-key, no one knows it's there, which is the whole point, and therefore it's effective.

AD: *What would be an example of a useful background sound?*

JM: There's all sorts of things to do with network activities. We've just prototyped a system that tells you that the server's being accessed; it could tell you that your boss has just sent you an AppleLink, that the printer's out of paper. You could have a scrolling message that comes up on screen, but it'd be nice if the computer just made an appropriate little sound and let me keep doing whatever I was working on.

AD: *If you had a drop box on the network and somebody put a file in it, it'd be nice to be told that, so you don't have to check the drop box every hour.*

JM: They're what I call subtle, meaningful things. They're not earth-shattering breakthroughs, but they're lower-level things that can make a very complete user experience. ♣

This is part two of a two-part article. The first part appeared in the December 1993 Apple Directions.

Sound: The Final Frontier

Sound Manager 3.0 Provides Industry-Best Sound

By Paul Dreyfus
Apple Directions *Staff*

Sound may be the great unexplored territory on the Macintosh computer, as Joy Mountford suggests in "State of the User Experience, Part Two" on page 13. A completely rewritten version of the Sound Manager gives you every reason to build sound into your applications (and no excuses not to). Sound Manager 3.0 makes sound playback on the Macintosh computer second to none in the personal computing industry.

Sound Manager 3.0 accomplishes this by providing a great many enhancements, including

- two to three times performance improvement
- 16-bit sound support
- the ability for users to select different audio playback devices
- support for plug-in compression/decompression software for any compressed audio format
- additional features and fixes to many bugs

The new version of the Sound Manager gives your Macintosh products immediate access to advanced sound capabilities; if they support a previous Sound Manager release, you don't have to alter them one byte to give your customers state-of-the-art computer-generated sound. In contrast to MS-DOS PCs, which often require additional coding to handle compression, decompression, rate conversion, mixing, and final output of sound, the Macintosh sound architecture takes care of most of these aspects of creating sounds.

Sound Manager 3.0 also offers special opportunities for developers of sound output cards and devices and compression/decompression software. (More on that soon; also, see the text box "Sound Manager 3.0: Key Messages and Opportunities," on this page.)

Make Sound Independent of Hardware

Before I tell you more about Sound Manager 3.0, there's a crucial detail about Macintosh sound that you need to know. When we released Sound Manager 3.0, Apple Computer, Inc., also made a significant change to the Macintosh sound architecture: The Macintosh Quadra 660AV and 840AV computers don't emulate the sound hardware used in the original Macintosh computers and emulated through the Apple Sound Chip (ASC) and other means ever since. Forthcoming PowerPC processor–based Macintosh computers also won't support the old sound hardware architecture.

This means that some of you may need to rebuild the sound code for your new products (and

Sound Manager 3.0: Key Messages and Opportunities

There are four specific messages you need to remember about Sound Manager 3.0.

1. Make your sound code independent of hardware by writing directly to the Sound Manager API (application programming interface) as defined in *Inside Macintosh,* Volume VI, Chapter 22 and the forthcoming *Inside Macintosh: Sound* (to be released in early 1994). If your sound code is written to the hardware specification outlined in *Inside Macintosh,* Volume III, Chapter 2, you'll have to rewrite it to the Sound Manager or it won't work (and may crash) on the Macintosh Quadra 840AV and Quadra 660AV computers or on forthcoming Macintosh with PowerPC computers, which don't use the old hardware solution for generating sound.

If you go to the trouble of rewriting hardware-dependent sound now, you not only guarantee your software compabitility with future Macintosh technology, but also give your customers access to the state-of-the-industry sound provided by the Sound Manager 3.0 if it's present on their computers. To be sure it's there, you may want to consider licensing Sound Manager 3.0 to ship with your product; to do so, contact Apple Software Licensing (AppleLink: SW.LICENSE or phone: [408] 974-4667).

2. Sound written to previous versions of the Sound Manager will have immediate access to the new Sound Manager's improved playback capabilities.

3. If you develop audio output devices, you have an oppportunity to develop new products that give users unprecedented computer-generated digital sound quality. You'll need to develop an add-on card for many new output devices and write a sifter for the device; for more information, see *How to Write a Sound Output Sifter,* which tells you how to write the software required to get sound from the Macintosh to external audio devices. If you're a member of Apple's Partners and Associates programs, you can obtain this publication by sending an AppleLink message to DEVSUPPORT. You'll also want to license the Sound Manager 3.0 to ship with your product.

4. There is a unique opportunity to write cross-platform compression/decompression software modules (called *codecs*) that give the Sound Manager the information it needs to support DOS/Windows compression formats. These codecs will be extremely useful to developers porting games and other products from DOS/Windows to the Macintosh environment; instead of having to use sound resampled and compressed into a new format, the ported application will play its original sounds on the Macintosh by means of the codec and the Sound Manager. For general information about codecs, see *Inside Macintosh: QuickTime Components,* available from APDA or your local bookseller; specifics about writing sound codecs are contained in *How to Write a Sound Output Sifter.* ♣

revised versions of your existing ones). Previously, some of you—especially in the games business—boosted performance by writing sound code directly to the hardware specification (outlined in *Inside Macintosh,* Volume III, Chapter 2), instead of to the Sound Manager. That approach is no longer necessary because of the performance made possible by Sound Manager 3.0; it also just won't work. Code that uses the hardware Gestalt routines won't play on the new Macintosh AV computers or on Macintosh with PowerPC computers, and it could break on them.

I'm sure this is a message that many of you won't like, but any existing applications with code written to the hardware need to be rewritten to employ the Sound Manager API (application programming interface) if they're to work on Apple's new hardware platforms. Put more positively, Apple is asking you to do the same thing with sound that you're doing with the rest of your software: make it independent of the hardware so it can easily be ported to the Macintosh with PowerPC platform.

The silver lining in this cloud is that Sound Manager 3.0 is backward compatible with previous versions, although it won't work with Macintosh 128K, Plus, Classic®, and SE computers. Any applications that use Sound Manager 3.0 will run on existing Macintosh computers (other than the ones just mentioned) as long as they are installed with the Component Manager, which is available as part of QuickTime 1.6 and System 7.1. If you have to rewrite portions of your software to make sound code "forward compatible" with future Macintosh hardware, that effort will pay off in a version that can give users of existing computers access to the Sound Manager's enhanced performance and new features.

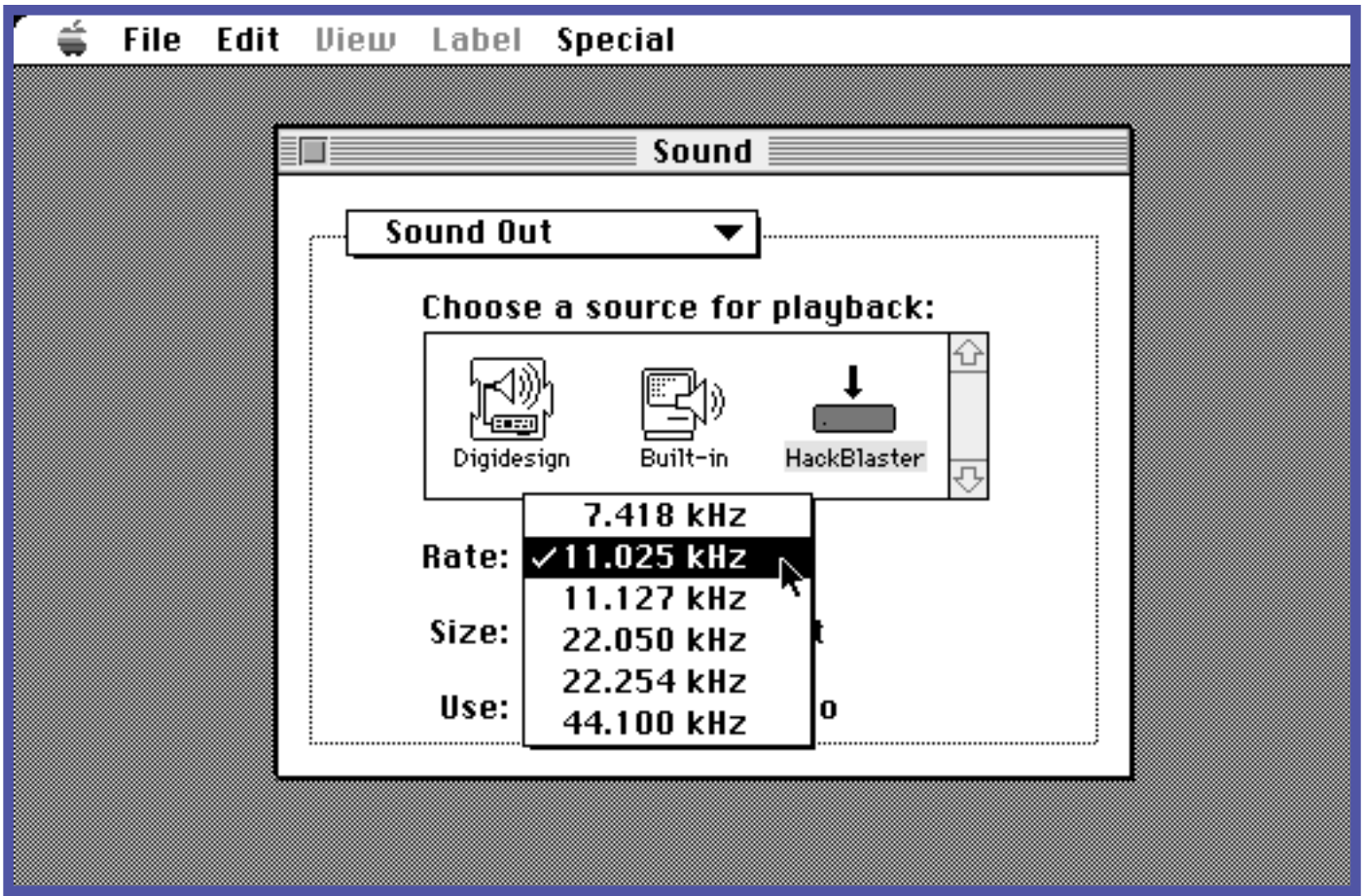
If your applications' sound code supports a previous version of the Sound Manager (and doesn't "hit" the hardware), it doesn't have to be altered to take advantage of the new sound enhancements, whether they're run on an older Macintosh computer that generate sound using the original hardware architecture or the Macintosh AV and forthcoming Macintosh with PowerPC computers.

New and Improved Features

For most developers, the opportunity offered by Sound Manager 3.0 is immediate improvement to the quality of sound and greater user choice when it comes to sound playback.

QuickTime developers and users will also appreciate the improvements Sound Manager 3.0 makes to QuickTime movie playback; QuickTime 1.6 automatically checks for the presence of Sound Manager 3.0 and, if the manager is installed, uses all its features.

Overall sound performance has been made more efficient and faster; Sound Manager 3.0 also improves sound quality on the Macintosh. Its two-to-three-times greater efficiency means applications can play more sounds at the same time and do other work while sound is playing; for example, a Macintosh LC can now play four channels of sound at once, a previously impossible feat. Greater sound efficiency also



The new Sound control panel, offering users the ability to select different sound output devices and playback rates.

results in increases in QuickTime movie playback frame rate.

Sound Manager 3.0 makes it possible, for the first time, for Macintosh computers to play 16-bit sound samples, making top fidelity a reality on the Macintosh AV computers and Macintosh with PowerPC computers, which will employ 16-bit sound. It will also automatically convert 16-bit samples to 8-bit samples for playback on Macintosh computers with 8-bit audio hardware. In addition, thanks to Sound Manager 3.0, the Macintosh computer can now provide true CD-quality audio with sample rates of up to 65 kHz, both in QuickTime movies and the audio portion of applications.

Sound Manager 3.0 also improves the sound quality of QuickTime movies and other sounds by using a faster linear interpolation for sample rate conversion from 11 kHz to 22 kHz, which makes audio samples at 11 kHz sound much better.

Sound Manager 3.0 makes a variety of other improvements that users will appreciate. You can now set left and right volume controls independently of each other. Additionally, the manager converts stereophonic sounds to monophonic for playback on mono hardware such as the Macintosh LC; previous versions of the Sound Manager simply dropped the right channel of stereo sound. In addition, it improves multiple-channel playback.

A New Opportunity

Another vast improvement is that users can easily direct sound to any available audio device through the new Sound control panel, which also lets users select hardware playback rates (see illustration).

Apple is encouraging audio device manufacturers to develop sound cards that enable users to record and play using different external devices with specialized features, such as sample rate conversion and audio mixing. If you'd like to develop a product in this category, you'll need the Apple publication, *How to Write a Sound Output Sifter,* which tells you how to write the software required to drive audio devices. Apple Partners and Associates can obtain it by sending an AppleLink message to DEVSUPPORT.

Cross-Platform Codec Opportunity

Another improvement some developers will want to take advantage of is the Sound Manager's ability to work with plug-in audio compression/decompression software (called codecs) using any format. Previous versions of the Sound Manager supported only MACE audio compression at ratios of 3:1 and 6:1. Now, if a codec for your compression format or ratio is installed in the user's System Folder, Sound Manager 3.0 has all the information it needs to decompress and play sounds recorded in that format.

QuickTime 1.6 takes advantage of this new feature as well by querying Sound Manager 3.0 for information on new compression types; assuming the right plug-in audio codec is available, QuickTime can then play compressed audio of any type.

This new capability will be particularly useful for those of you porting DOS/Windows products, which use proprietary compression software, to the Macintosh computer. Previously, the Macintosh computer would not have recognized the compression format, making porting an expensive proposition requiring resampling and recompressing of sound in MACE 3:1 or 6:1 format. Now, an audio codec will give the Sound Manager the information it needs to support any compression format.

If I were a developer who knew a lot about compression technology, I'd get to work today on cross-platform codecs that let the Macintosh play sound using DOS/Windows compression formats. More than one developer will be interested in licensing your product to make it less expensive to port games and other multimedia software to the Macintosh platform.

Bug Fixes, Other Improvements

One of the major reasons Apple engineers took on a rewrite of the Sound Manager was the number of bugs—some of them major—that previous versions contained. They're glad to report that they've fixed a great many of the bugs, and that one vast improvement you'll find with Sound Manager 3.0 is its robustness and stability when compared with older versions.

For a list of the specific bugs that have been fixed as well as the many other minor improvements made to Macintosh sound, see the article "What's New With Sound Manager 3.0," *develop* Issue 16, page 34, or the Sound Manager 3.0 Tech Note on the Developer CD. (On the November CD, *Northern Hexposure,* you can find the Tech Note by following the path Dev.CD Nov 93:New System Software Extensions: Sound Manager 3.0.)

If you haven't already checked out Sound Manager 3.0, you'll want to do so soon; you can find it on the November Developer CD at the same location as the Tech Note just mentioned. (The Sound Manager Developer's Kit version 3.0, which includes the manager and the Tech Note, is also available through APDA [R0507LL/A]. See page 28 for APDA ordering information.)

You'll probably want to do what a lot of people do when they first install the new manager: try out the "Simple Beep" system alert sound. It sounds different than what you're used to hearing; that's because the Sound Manager 3.0 square-wave synthesizer produces true square waves, unlike previous versions that produced something more like a modified sine wave. So the Simple Beep now sounds the way it was supposed to in the first place.

Once you've played, you'll want to get to work making sure your code is written to the API and not to the original sound hardware. A small investment of time today will be worth a lot tomorrow, in sales to both the future and past Macintosh installed base. And you'll be paving the way into one of the unexplored frontiers on the Macintosh computer. ♣

CD Highlights

continued from page 11

article "International Number Formatting" in *develop* Issue 16, but if you know that your application will only have to work with system software localized for the same language, the localized number format descriptions are usually adequate. And this ResEdit editor is certainly the nicest way to create the localized resources.

The FMAT editor includes these features:

- It creates new 'FMAT' resources from a format string.
- It lets you pick characters for format strings from a palette.
- It works with any script system.
- It displays sample numbers to give you immediate feedback.

Macintosh Tech Notes Updates

This folder contains new and updated Tech Notes and Q&As for January 1994, including OV 20 - Internationalization, TB 09 - Finder Flags, TB 14 - MultiFinder FAQ, and TB 575 - Window Mgr Q&As.

Sample Code Survey

The Developer Support Center is currently evaluating available code samples and defining new standards for presentation. We are soliciting developer feedback, so that we can provide you with the samples you need, or improve the samples that exist now.

The Sample Code Board has devised a survey that we hope will provide us with information about which samples developers recognize as valuable and which obsolete, which technology areas need more coverage and which are over-represented, which levels of expertise need more attention, and so forth. Please fill this survey out and return it to DEVSUPPORT.

System 7 Pro

This folder contains the following components, which are intended for AOCE developers:

- background information on AOCE technology
- System 7 Pro software
- AOCE interface files
- a problem fix for developers using high-level debuggers
- final and proposed Apple event suites

AOCE developers should obtain the AOCE Software Developer's Kit (SDK) from APDA. The AOCE SDK includes

- System 7 Pro software
- software updates to PowerTalk (for certain developers)
- AOCE interface files
- AOCE sample code
- AOCE development tools
- PowerTalk user documentation (on CD only)
- AOCE development documentation (on CD and paper)

In addition, developers interested in the following items should purchase the System 7 Pro Upgrade Kit:

- voucher for direct dial-up mail access software
- voucher for free DigiSign signer
- printed user documentation
- version of System 7 Pro with network encryption (see below)

System 7 Pro Software. The "developer version" of PowerTalk included on this disc does not include network encryption capability.

This special version is different from the standard version (version 1.0). It includes a version of ASDSP that does not encrypt network communications. However, the two versions can be used together; even though users can tell that something sent from or to a user of the special developer version was not sent encrypted, the difference is transparent to applications. The encryption technologies used by the PowerTalk "key chain" and DigiSign are not affected in any way.

PowerShare Software. This CD does not include PowerShare Collaboration Servers. If you're planning to build server-based Message Service Access Modules, you should purchase the PowerShare software from an Apple authorized reseller.

Software Updates

A problem that affects developers using certain high-level debuggers was discovered; it's been fixed by an application that patches AppleMail. Note that redistribution of this update is not permitted.

Other Updates

Any updates to documentation, interfaces, and so on, will be posted to the appropriate areas on AppleLink. Be sure to check AppleLink on a regular basis.

VUMeters

This application displays VU (volume unit) meters, much like the ones found on tape decks. You can use it to monitor the sound input and output channels on the Macintosh AV computers to ensure that the proper sound level is present.

The VUMeters application can display sound levels both as peak readings and as more conventional power readings. You can select whether you want to measure levels in terms of the analog voltages at the connectors (the conventional way) or in terms of the peak level of the computer's A/D and D/A systems. The application conforms as much as possible to the standard for VUMeter dynamics.

This package is useful as a debugging aid when working with sound. You can also use it as a complete example of how to develop code for the Apple Real Time Architecture (ARTA) DSP system on the new Macintosh Quadra AV systems.

Coming Soon

Next month's Tool Chest CD will feature an overhauled Testing & Debugging folder; in addition, we hope to have more Developer Notes, Macintosh with PowerPC information, and the Chinese Language Kit. See you there!

Alex Dosher
Acting Developer CD Leader

Business & Marketing

Inside This Section

Apple Directions, the Book, Now Available

Yes, you read that correctly. *Apple Directions,* or at least highlights of its Business & Marketing section, has become a book, *The High-Tech Marketing Companion.*

Developed and edited by *Apple Directions* Business & Marketing Editor Dee Kiamy and published by Addison-Wesley, *The High-Tech Marketing Companion* gives you the same kind of practical, how-to marketing advice that you're accustomed to reading every month in our Business & Marketing section.

The High-Tech Marketing Companion is a collection of articles previously published as Developer Outlooks and Marketing Features on the pages of *Apple Directions.* Some are written by developers themselves, and others are penned by industry experts who are recognized in their respective fields.

Apple publishes *Apple Directions* to help you maximize your development dollar; *The High-Tech Marketing Companion* has the same purpose. It can help you make better-informed business and marketing decisions and apply your often limited resources in the most efficient way possible, helping you succeed in an increasingly competitive environment.

We'll be telling you a lot more about *The High-Tech Marketing Companion,* and where you can obtain it, in next month's issue of *Apple Directions.* The book is now available through your local bookseller. Addison-Wesley will also be offering it at their exhibit at Macworld San Francisco January 5–8. ♣

Market Research Monthly

Configuration Data, Part One: Color by the Numbers

Among the factors you need to think about when you're designing new products or updating old ones is how customers in your target market set up their Macintosh computers—for example, whether they use monochrome or color monitors, the size monitor they use, how much RAM they've installed, and the amount of storage capacity their systems have.

The next several installments in Market Research Monthly provide configuration data about Macintosh computers that Apple Computer, Inc., gathered in its latest study of Macintosh customers worldwide, completed in spring 1993. Apple collects this data to track trends in the installed base and help make decisions about what new products to offer. We're providing you with this previously unreleased information to help you with your product planning process.

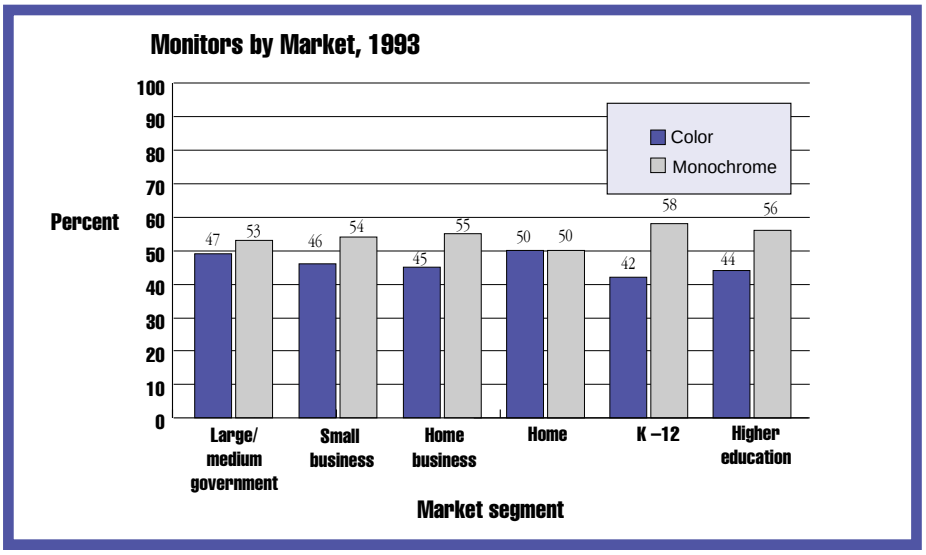
Color vs. Monochrome

Not so long ago, you had to decide whether it was worth the resources to build color capabilities into your product; now, color has so much invaded the hearts and minds of computer users, not to mention the Macintosh installed base, that many of you are wondering if it's worth making your products look good on monochrome monitors anymore.

Apple's customer research suggests just how important it is for you to support color with your applications, as Apple has been urging you to do for some time. However, a large percentage of Macintosh customers still use monochrome monitors; some of you, depending on your product and market, still need to think about how your product looks in black and white.

Today's Users

Looking at the current installed base, more than half of all Macintosh users around the globe still use monochrome monitors with their desktop systems. This is true of every segment of the global Macintosh market, as shown in the figure "Monitors by Market, 1993."

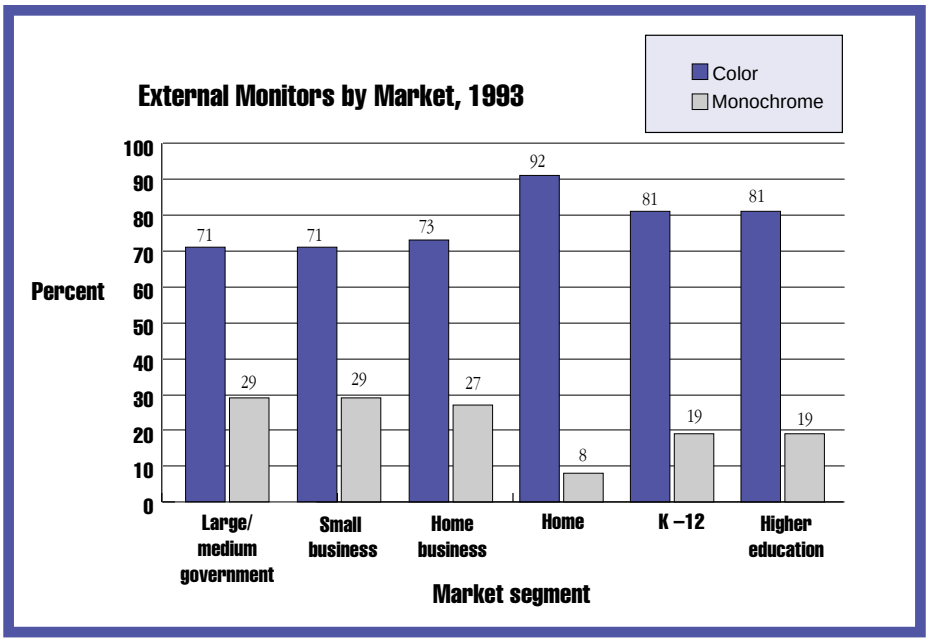


As that figure shows, the market segment most likely to have chosen color is nonbusiness users in the home, 50 percent of whom have color monitors.

These numbers are weighted by the fact that nearly half of the monochrome monitors are the 9-inch displays built in to the Macintosh Plus, SE, SE/30, and Classic computers. Since Macintosh computers with a built-in 9-inch monochrome display are no longer sold by Apple, this chunk of the market is likely to dwindle over time.

A Glimpse of the Future

"External Monitors by Market, 1993," which looks only at external monitors not built in to the so-called classic-style models, provides a glimpse of the future of the Macintosh market. It's expected that the older models that use 9-inch monochrome monitors will increasingly be retired in favor of more powerful systems that support color, especially as Apple's current high-volume sales strategy continues its current success. (Apple sold more new systems in the fourth fiscal quarter of 1993—July through September—than it sold in any previous quarter, and the vast majority of those systems were equipped with color monitors.)



The Spring 1993 customer data shows that black and white is on the way out: Less than 30 percent of each market segment uses an external monochrome monitor. In the nonbusiness home market, only 8 percent use external monochrome monitors. External monochrome monitors are used most extensively in large, small, and home business settings (which tend to employ larger monitors than do other market segments).

That's the story about color and monochrome monitors in the Macintosh customer base. In the coming months, Market Research Monthly will look at monitor sizes, amount of ROM, storage capacity, and other pertinent configuration data from Apple. ♣

Marketing Feature

Building Better Documentation

By Molly Tyson,
Apple Instructional Products

Most people don't curl up in front of a fireplace with a cup of hot tea and a comforter to read manuals for pleasure—they read them for information about how to use a product. Think of yourself when you're learning to use a new software product—or assembling a new lawn mower, for that matter. If you use the documentation at all, you're looking for answers to questions—preferably in pictures with numbered callouts telling you what to do first, second, and third. How do I turn it on? How do I install it? How do I get it to print? Or, in the case of the lawn mower, how do I attach the big cable to the round thing? And what do I do with this extra bolt?

The most effective documentation comes from companies that treat their tutorials and manuals as extensions of their product's user interface. Well-designed documentation makes it easy for beginners to become productive quickly and for experienced users to take advantage of the full functionality of your product. It can make a product easier to use and therefore more successful in the marketplace.

At Apple Computer, Inc., documentation groups work closely with interface designers and other engineers to create the best possible user experience for our customers. Our efforts have been rewarded with best of show awards in competitions sponsored by the Society for Technical Communication and with consistently high rankings in JD Powers customer satisfaction surveys. This article presents an opportunity to share our philosophies and advice on this important topic. So, whether your company is developing its first or fiftieth product, we hope you'll be able to take away a few ideas for making your manuals, computer-based training, and on-line help systems better.

Good Documentation Is Good Business

Good documentation is good business, but quantifying the benefits is often difficult. Unfortunately, it's easier to see the added project development costs and the ongoing burden to your cost of goods sold (COGS). Great documentation needs to be "sold" to the company as an investment in two different areas: customer support and customer satisfaction. First, effective training and reference information results in users quickly finding what they need in their documentation, so they're less likely to call your customer support line. Second, good instruction makes the product easier to use and enhances user productivity. If users can teach themselves how to use your product, it saves their corporation training and support costs. Bottom line, this enhanced productivity gives you a competitive advantage in the marketplace, and these satisfied

customers are more likely to upgrade and buy your follow-on products.

Developing Cost-Effective Documentation

Effective documentation doesn't have to be expensive. One way to reduce documentation-related costs is to take a minimalist approach—write lean. Users aren't interested in reading lengthy paragraphs of explanatory text. They go straight to the step-by-step instructions and illustrations.

You can keep development costs down by leveraging information from one product's documentation to another or by designing instructional materials so that one book is common to all products in a product line. This also saves time for localizers, because they only have to translate the material that's different from one product to the next.

Another important way to control costs is to have writers work closely with engineering groups and link documentation schedules to engineering milestones. When writers work closely with engineering groups, bugs tend to be corrected, so writers don't have to document elaborate "workarounds." This saves pages while improving the user experience. Linking documentation schedules to engineering milestones keeps documentation off the project's critical path and prevents it from being finished prematurely, while product features are still changing. Throwing inaccurate books away and paying twice for printing may not be as costly as delaying the introduction of a product, but these are expenses you can avoid if you keep documentation and product development in step.

So much for theory. Read on for detailed advice about creating better tutorials, manuals, and online help systems.

Building Your Team

Good documentation starts with good instructional designers—writers trained to create teaching tools. Asking engineers or marketers to write the documentation is a poor use of their time, and the resulting documentation is likely to frustrate users, lower sales, and increase customer support costs. Good instructional designers know how to organize and filter information so that users get what they need without being overwhelmed. They're trained to think like users and explain things in a way that makes sense. Really good instructional designers also save you money by designing minimalist instruction. (It's actually harder to write lean than to write long.)

Developing great documentation involves other specialists: editors, art directors, illustrators, interface designers, and animators.

Editors improve documentation by identifying missing topics, addressing inconsistencies in tone, clarifying confusing instructions, and suggesting ways to reword problem sentences.

Art directors define the most effective use of visual elements and text and help users navigate thrugh instructional materials by careful planning of layout, art, and typography.

Illustrators provide art that is better than words to show certain procedures and concepts. (And illustrations developed for user manuals can be reused for advertisements and marketing collateral.) Interface designers work with instructional designers to plan the interaction between the computer and user so that navigation is easy and feedback is meaningful. Animators make storyboards come to life by putting the text and graphics into a scripting environment and programming the interactions between the computer and user.

Planning Your Documentation

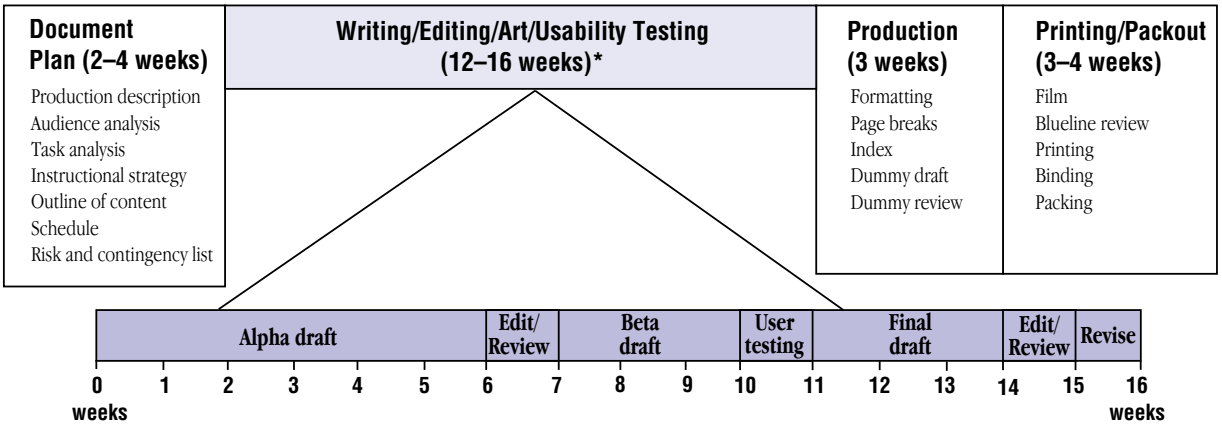
Planning your instructional strategy before developing your documentation is like prepping a house before painting it. It's time consuming, but in the long run it leads to better results and saves time by streamlining the rest of the development process.

Make sure your instructional designers attend engineering project meetings as early in the product development process as possible. Review the documentation plan with the engineering team, and set expectations on when they'll be required to review documentation.

Develop a documentation timeline like the one shown on this page to help coordinate your documentation schedule with the engineering schedule. This timeline will vary depending on the scale of your project, but it's a reasonable estimate for most manuals.

In some cases you can save time by spending more money on resources. Your instructional designers can advise you on where the schedule can be compressed and where it can't. For example, it may not save you any time to throw extra writers on a project at the end, because of the time it takes them to get up to speed. And

Sample Timeline



*Development time is estimated for a 200-page manual and depends on the complexity of the product, volatility of technology, access to prototypes and subject matter experts, and timely feedback from reviewers, among other factors.

all the money in the world can't get a printer to print and bind a 200-page manual in less than a week—this is a complex, multistep process that can't be rushed.

Doing a Task Analysis

The first step in planning your instructional strategy is to identify the tasks that a user must undertake to learn and work with a product. Formulating a task analysis requires that an instructional designer

- review documents related to the product (engineering specifications, marketing plans, manuals for similar products)
- interview engineers, product managers, testers, and others associated with the product
- use the product (once a functional prototype is available)
- observe the product being used by others

Create an outline or a tree chart of essential user tasks, then test it. Do this by observing an experienced user coaching an inexperienced person who is learning to use the product. Any missing tasks that come up in the experienced user's task descriptions should be added to the task analysis. If the product isn't yet functional, you can test the task analysis by having the product designer describe the steps a user would follow to use the product.

When you're designing products for use with the Macintosh or other Apple products, it makes sense to know what Apple has already documented. You can assume that the user is already familiar with any basic concepts covered in the Macintosh manual and go from there—saving dozens of pages in the process. And by using terminology that's consistent with terminology used in the "host" product, you'll reinforce skills and concepts users have already learned instead of making them learn a new dialect to use your program. (The *Apple Publications Style Guide,* available from APDA, provides standard Apple terminology that you can adopt.)

Analyzing the Audience

During the task analysis, you need to learn about the type of users who will buy the product. What do they already know about the product or subject matter? How will they use the product? Where will they use the product?

When you're familiar with the audience, you can provide meaningful examples to describe specific tasks. If the product is aimed primarily at the business market, you'll want to use business examples in your instruction. If it's aimed at students, you'll want to use educational examples. If it's aimed at a broad market, you'll want to choose examples that anyone can understand and relate to. The product manager is your best source of information about the target audience.

Determining Your Instructional Strategy

Once you understand what there is to teach and whom you're teaching, you can develop your product's instructional strategy. How will you organize the information, how will you present it, and how will you deliver it?

If setup is required, you'll need to provide setup instructions.

If the product is complex, you'll probably want to include a tutorial that guides users through the essential activities involved in using the product and a reference that provides comprehensive information about using the product. If the product is simple, you may need only a brief overview explaining what the product can do and how to use it. Test your assumptions by watching people use early drafts of your documentation.

Organizing Information for Different User Needs

As you're developing your instructional strategy, keep in mind that different people need different kinds of information.

Beginners need a lot of context setting and hand holding. They may or may not know what they want to do with the product.

Experienced users often have particular goals in mind and want

Learn by Doing

Computer-based training (CBT) can be lots of fun. Grown men and women laugh out loud when they click the fish bowl in Macintosh Basics and bubbles come out of the fish's mouth. They're surprised and delighted when the coffee cup steams, the clock whizzes, and staples fall out of the stapler. People don't expect the elements on the screen to respond to their actions in such a playful way. They expect computers to be serious, and discovering that learning can be fun helps to defuse anxiety. CBT seems to be most fun if the elements on the screen are engaging and cartoon-like rather than realistic.

CBT lets you focus people's attention on what you want them to learn and protect them from making mistakes that would be confusing and demoralizing. You can do this by disabling menus that aren't in use and taking commands out of menus to shorten them so they don't obscure text on the screen.

We follow very simple design principles in developing CBT for Apple products: motivate, provide plenty of opportunity for practice, provide remediation when the user wants it, keep the navigation simple, provide opportunities for users to test their knowledge, sequence the training so that users master prerequisites before going on, and do lots of user testing.

The greatest strength of CBT is that it can protect new users from becoming frustrated. At some point, users need to be trained in the real interface, but new users really benefit from a successful experience in the safety of a protected environment.

—Sandy Korzenny
Instructional Designer
Apple Computer, Inc.

answers to specific questions. They start "reading" from the back of the book—with the index—and go directly to the information that pertains to what they're doing. They may also be curious about what else they can do with the product.

Dividing information into setup, training, and reference sections gives all users what they need. Beginners can get the information they need without encountering complex information that's over their heads, and experienced users can get the procedural information they need without being slowed down by detailed instructions for novices.

Choosing the Media

Once you've mapped out an instructional strategy, you need to decide how to deliver the information. For each documentation component, your choice of media should be determined by a combination of factors—the user's needs, your budget and schedule, packaging constraints, and your expertise developing in different media.

For setup instructions, consider

- a numbered list with illustrations showing each step in the setup procedure
- a poster showing the entire process on one page
- a video showing somebody assembling the product

For tutorial information, consider

- computer-based training that provides guided practice with feedback, and sound and animation to help motivate new users
- a printed tutorial that guides the user through practice sessions with the real product and serves as a "how-to" reference later on

For reference information, consider

- a printed reference manual that provides comprehensive information about the product and a detailed index
- an on-line reference system (if you can engineer it so that users can get answers to questions without leaving the application)

Writing a Documentation Plan

After you've figured out your instructional strategy and teaching approach, you should write and distribute a documentation plan.

The documentation plan should include

- a brief description of the product
- a description of the target audience
- an explanation of your instructional strategy, including the learning path that explains how different users would find their way through the instructional materials
- an outline of topics within each instructional component
- a schedule (linked to milestones in the engineering schedule) for each component
- a description of any risks and contingencies

Writing Setup Instructions

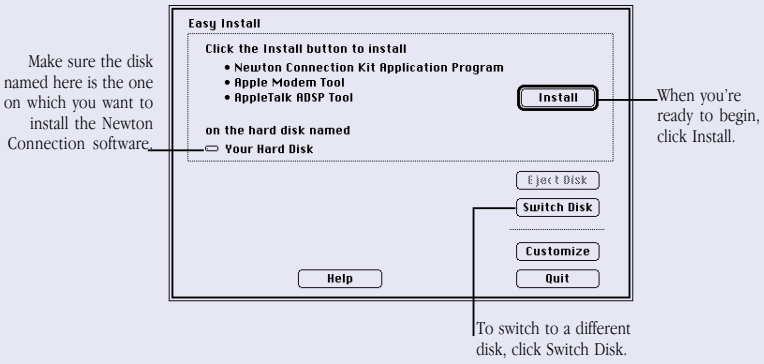
Setup instructions should be the first thing customers see upon unpacking the product. Here are some tips for writing effective setup instructions:

- Illustrate all the steps involved in setting up the product.
- Limit each step to one action.

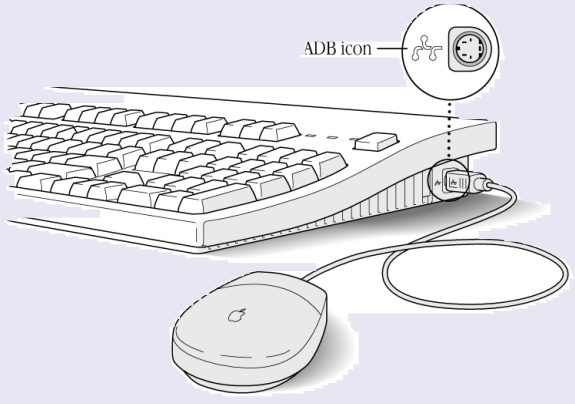
Using Art to Communicate

Using art to communicate technical information is often more effective than using words. In usability tests it's common to see people flipping through books looking for pictures that will show them what to do. They start "reading" with the illustration and resort to text explanations only if they can't get what they need from the art.

Recognizing that this is how people like to get information, we rely on illustrations or screen shots with callouts (explanations with pointers to different parts of the illustration or screen) to communicate much of the information in Apple manuals. ♣



Callouts make illustrations even more informative. (Too many callouts can be confusing. Treat them as design elements as well as instructional elements.)



Insets help a user "zoom in" on important details in a drawing and help locate the scene of the action.

(People like to check off each step as they complete it; if two actions are put in one step, users may miss the second action.)
- Test your instructions on a person who is unfamiliar with the product. Make notes about the steps that were confusing, then retest your revised instructions on another person who is unfamiliar with the product.
- Put warnings *before* the activity they pertain to.
- Tell people what to do when they've finished setting up the product.

Writing a Tutorial

A tutorial guides the user step by step through essential activities in an application program. It introduces a sampling of commonly used functions in the context of a useful activity.

Don't try to teach everything about the product in the tutorial. Focus on the main features of the product. Your goal is to make users confident and competent enough to learn on their own—through exploration or from the reference section of the book.

Write your tutorial with the least experienced users in mind. Tell them what they're going to do, how to do it, and what to do if something goes wrong. Show them what their screen will look like at various stages. Reassure them that they're on the right track.

Any information covered in the tutorial should also be covered in the reference section of the manual. (You can't assume that experienced users will read the tutorial, and you don't want to force users who did use the tutorial to return to it for the facts that were woven into it.)

Keep the following guidelines in mind as you write the tutorial:

- *Focus on "survival skills."* Avoid overloading new users with too much information.
- *Put the learning in a context.* Why and when would a user do what you're teaching?
- *Explain what to do in a series of boldface numbered steps.* If necessary, give additional information about the procedure below the numbered step. But keep your discussions as brief as possible; your emphasis should be on doing rather than reading.
- *Use art to reassure users that they're on the right track.* If possible, place the art next to the information it relates to.
- *Design your tutorial so it can be completed in 30 minutes.* You may want to present several lessons that together last two or three hours, but each lesson should take no longer than 30 minutes.
- *Be accurate.* Mistakes are unfortunate anywhere in a manual, but they are deadly in a tutorial, because each section builds on the one before. If you lose your readers in step 3, you've lost them for the rest of the tutorial. You've also lost their trust.

Designing Computer-Based Training

The design principles for computer-based training (CBT) are the same as for print-based training, but you have many more tools at your disposal. You can use text, graphics, sound, animation, and interaction to communicate information. And you can provide a more protected environment for learning than you can with a print tutorial. By programming the computer to trap errors, you can prevent mistakes and provide appropriate remediation. Mistakes become learning experiences instead of phone calls to customer support. (See the text box "Learn by Doing" for more on designing effective CBT.)

Writing a Reference Manual

The reference section of your manual should explain everything there is to know about the product or program. Keep in mind that few people will read the reference section sequentially, so each task or explanation should stand alone and should be easy to find. Pay particular attention to developing the table of contents and index.

The way you organize reference material into chapters depends on what you're writing about. If activities must occur in a particular order, you should present them in their logical sequence. If activities fall into logical categories, you might assign a chapter to each category and order the activities within each chapter according to their frequency or importance.

Starting with your table of contents will help you figure out a logical arrangement for your information. Looking at manuals on similar subjects will give you an idea of how others have solved (or failed to solve) similar organizational problems.

In general, organize your manual around the tasks the user bought the tool to perform, not around what each tool or command can do—think like a user rather than an engineer.

Developing On-line Help

Some on-line help systems are merely electronic versions of the reference manual. A more useful alternative is to design help that interacts with the users (presenting information in words familiar

On-line Help

The most important thing to keep in mind when you're writing on-line help systems is that you're working with a new medium that requires a different instructional approach. You aren't constrained to a one-track, linear learning path, and you have the tools to create an interactive experience—one that can be adapted and customized every time it's used.

An effective way to design this interaction is to think of yourself as a personal coach, watching over users' shoulders and guiding them through the steps required to solve a problem. What information do they need to successfully accomplish the next step, and what questions might arise?

Present instructions one step at a time, providing breaks that let the user do what you've just explained before moving on. Keep text very clear and succinct; extraneous information can be confusing. Use a direct, active voice. Don't overload the main sequence with definitions and related concepts. Make additional information available as an option that users can pursue if they need it.

Use context checking to adapt your instructions to fit the user's situation. Skip unnecessary steps whenever you can, or add additional steps as needed. Check that users are staying on track, and when they make a mistake, give immediate feedback and corrective guidance. You're no longer limited to a single instructional path that must be designed to fit most cases. Make sure that the information presented is always relevant to your users.

Finally, stay focused on providing users with just enough information to help them solve their problems so that they can get back to work. Use other forms of documentation for comprehensive reference materials. On-line help is very effective, but it's just one piece of a complete instructional delivery strategy.

—Glenn Katz
Human Interface Engineer
Apple Computer, Inc.

to users and letting them choose what they want to see) and makes use of the context in which they are working. Such a help system requires instructional designers to plan guided interactions that walk users through tasks, displaying explanatory text and graphics as necessary. This type of interactive help is more costly to develop than a straightforward on-line reference but can be more useful.

(See the text box "On-line Help" for more on designing effective on-line instruction.)

Writing Tips

Whether you're working on a tutorial, a reference manual, or on-line help, a few important writing tips will help make your instructional writing more engaging and effective:

• *Use the second person.* Addressing the user as "you" has two advantages: It's informal, so it puts the reader at ease, and it forces you to write in the active rather than the passive voice.

• *Use examples.* Put yourself in the reader's shoes when you think up examples. It encourages you to write about how to use the product rather than just writing a product description.

• *Use consistent terminology and avoid jargon.* It's hard enough to learn a new product without learning a foreign language at the same time.

• *Get the user to do something right away.* Don't spend a lot of time on introductory material. Start the tutorial and let the user use the product as soon as possible.

• *Support different learning styles.* Some people learn best through guided tutorials. Others like to learn by doing their own work. Accommodate all kinds of learners by covering the same material in different ways: on training disks, in reference chapters, in on-line help, and on quick reference cards.

• *Use lots of art.* Illustrations are often easier to understand than explanations. Screen shots reassure users that they're in the right place. It's hard to overdo it on graphics. For more on using art to enhance instruction, see the text box "Using Art to Communicate."

User Observation

User observation simply involves watching and listening carefully to users as they work with your documentation. Although it's possible to collect far more elaborate data, observing users is a quick way to obtain an objective view of your documentation. Repeated exposure to testing helps instructional designers anticipate users' documentation needs.

Preparing for a User Observation

• *Set an objective.* Before you do any testing, determine a test objective that focuses on a specific aspect of the documentation. By limiting the scope of the test, you're more likely to get information that will help you solve a specific problem.

• *Design the tasks.* Your test participant will work through one or more specific tasks. These tasks should be real tasks that you expect most users to carry out when they use your product. The entire user observation should not run over an hour, so you should design tasks that focus on the part of the documentation you're studying. After you determine which tasks to use, write them out as short, simple instructions.

• *Find representative users.* When looking for participants, try to find people who have the same experience level as a typical user of your product. Don't ask people you work with regularly to be participants because they'll probably be familiar with your product or your opinions about the product.

Conducting a User Observation

• *Describe the purpose of the observation.* Set your participants at ease by stressing that you're not testing them—you're trying to find problems in the documentation. For example, you could say: "You're helping us by trying out this documentation in its early stages. If you have trouble with some of the tasks, it's the documentation's fault, not yours."

• *Tell the participant that it's OK to quit at any time.* Make sure you inform participants that they can quit if they find themselves becoming uncomfortable. Participants shouldn't feel like they're locked into completing tasks.

• *Explain how to "think aloud."* Ask participants to think aloud during the observation, saying what comes to mind as they work. By listening to participants think and plan, you'll be able to examine their expectations for your documentation, as well as their intentions and their problem-solving strategies. You'll find that listening to users as they work provides you with an enormous amount of useful information that you can't get in any other way.

• *Explain that you will not provide help.* It's very important that you allow participants to work with your documentation without any interference or extra help. This is the best way to see how people really interact with the documentation. For example, if you see a participant begin to have difficulty and you immediately provide an answer, you'll lose the most valuable information you can gain from user observation—where users have trouble, and how they figure out what to do.

• *Describe the tasks and introduce the product.* Explain what the participant should do first, second, third, and so on. Give the participant written instructions for the tasks.

• *Conduct the observation.* As you observe, you'll see users doing things you never expected them to do. When you see participants making mistakes, your first instinct may be to blame the mistakes on the participant's inexperience or lack of intelligence, but this is the wrong focus to take. The purpose of observing users is to see what parts of your documentation might be difficult or ineffective. Therefore, if you see a participant struggling or making mistakes, you should attribute the difficulties to faulty documentation design, not to the participant.

• *Use the results.* To get the most out of your test results, review all your data carefully and thoroughly. Look for places where participants had trouble, and see if you can determine how your documentation could be changed to alleviate the problems. Look for patterns in the participants' behavior that might tell you whether the documentation was understood correctly.

It's a good idea to keep a record of what you found out during the test. That way, you'll have data to support your design decisions and you'll be able to see trends in users' behavior. After you've examined the results and summarized the important findings, fix the problems you found and test the documentation again. By testing your documentation more than once, you'll see how your changes affect users' performance.

Kate Gomoll,
"User Observation:
Guidelines for Apple Developers"
Macintosh Human Interface Guidelines

• *Think like a user.* Put yourself in the user's shoes—and stay there. Observe other people using your documentation.

The Importance of Editing

Good organization and clear, concise writing are vital to the success of any instructional materials. Having writers review one another's work can help to identify problem areas, but there's no substitute for a good editor. Ideally, you should hire editors who are skilled at two types of editing:

• developmental or substantive editing—reviewing early drafts for problems with organization, focus, audience level, and tone

• copyediting or line editing—reviewing later drafts for problems with usage, grammar, spelling, punctuation, adherence to house style rules, and inclusion of all necessary legal and safety boilerplates

User Testing Is Essential

Usability testing should be central to your development process. Find users who match the target audience and watch them use your documentation. By asking them to think out loud, you can see what's confusing and get ideas for how to improve the instruction. If the schedule permits, test again to make sure the revision addresses the problems.

(See the text box "User Observation" for information on how to conduct informal usability testing.)

A Final Word on Documentation

Good documentation is a long-term investment that results in reduced customer support costs, increased customer satisfaction, and more repeat sales. The best and most cost-effective documentation comes from building the right team, executing a detailed plan, working in concert with engineering, and, most important, thinking like your users. ♣

Molly Tyson is manager of Instructional Products at Apple Computer, Inc.

Molly joined Apple in 1981 as an instructional designer. She has written more than 60 manuals for Apple products. Before becoming an instructional designer, Molly worked as a journalist. Her work has appeared in the New York Times, Self, Highlights for Children, Runner, *and* Popular Computing.

Now Available From Apple

The following list shows APDA products that have become available to developers within the last several weeks. To get a full listing of all APDA products, check the current *APDA Tools Catalog.* For new product announcements and the most up-to-date price lists, check AppleLink (path—Developer Support:Developer Services:Apple Information Resources:APDA—Tools for Developers).

Apple Tools

Apple Media Tool Programming Environment
R0535LL/A
$2,995

Apple Media Kit (this bundle includes the Apple Media Tool and the Apple Media Tool Programming Environment)
B1453LL/A
$3,495

AppleSearch Client Developer's Kit
R0468Z/A
$299

Newton Toolkit Version 1.0b7 (U.S. version) H0107LL/A
Newton Toolkit Version 1.0b7 (International version) H0107Z/A

HyperCard Version 2.2
M2365LL/A
$99 (special introductory price; regular price will be $249)

AOCE Software Developer's Kit
R0525LL/A
$195

PowerTalk Templates
R0530LL/A
$99

VIP-C Development System for Macintosh
T0822LL/A
$395

AppleScript Software Development Toolkit Version 1.1
R0175Z/B
$199

AppleScript Software Development Toolkit Version 1.1 upgrade
R0557Z/A
$99

MPW Development System Version 3.3.1 (CD-ROM)
R0505LL/B
$250

MPW Development System Version 3.3.1 (disk and CD-ROM)
R0503LL/B
$300

MPW C Version 3.3.2
M0022LL/H
$100

MPW C++ Version 3.3.1
M0346LL/H
$150

MPW Object Pascal Version 3.3.1
M0021LL/G
$100

MPW C Version 3.3.2 Update
M0325LL/H
$15

MPW C++ Version 3.3.1 Update
M0852LL/E
$15

MPW Object Pascal Version 3.3.1 Update
M0321LL/G
$15

ToolServer Version 1.1.1
R0449LL/C
$100

APDA Ordering Information

To place an APDA order from within the United States, contact APDA at (800) 282-2732; in Canada, call (800) 637-0029. For those who need to call the United States APDA office from abroad, the number is (716) 871-6555. You can also reach us by AppleLink; the address is APDA. If you're outside the United States, you may prefer to work with your local APDA contact. For a list of non-U.S. APDA contacts, see the "International APDA Programs" page in the *APDA Tools Catalog.*