The Developer Business Report *December 1994*

Apple Directions

Inside This Issue

New Newton Developer Programs Coming Soon

Apple Computer, Inc., will soon announce new support programs for Newton developers. Details weren't available as this month's *Apple Directions* went to press, but we'll describe the new programs in the January issue. In the meantime, check the Newton area of AppleLink on December 1 for a special message about the programs.

Apple News

New Hardware Reference Platform for PowerPC Processor

Open Platform to Run Mac OS, Other Operating Systems

Apple Computer, Inc., IBM Corporation, and Motorola Corporation recently agreed on a new hardware reference platform for the PowerPC microprocessor that aims to deliver a much wider range of operating system and application choices for computer customers.

The new platform defines an architecture—a framework made up of open technical concepts, definitions, specifications, and interfaces—that any hardware vendor can use to build compatible PowerPC processor–based computers that are ready to run a tremendous variety of applications written for a number of industry operating systems.

The new hardware reference platform for the PowerPC microprocessor is expected to support popular operating systems, such as the Mac OS, AIX, OS/2 for the PowerPC, and Microsoft Windows NT. Many of these operating systems already support current PC applications.

More importanty, to ensure that the vast majority of applications written to current

please turn to page 7

Strategy Mosaic

This Changes *Everything*

By Gregg Williams, Apple Directions *staff*

The Hardware Reference Platform for PowerPC, Apple's Licensing Strategy, and You

Whatever you're thinking about Apple, think again. Two recent announcements combine to change everything—not just for Apple Computer, Inc., but for you, too:

- Last September, Apple announced that it would license its operating system, now named *Mac OS,* to selected companies for use worldwide.
- Apple, IBM, and Motorola recently expanded the scope of Apple's licensing plans by announcing a new hardware reference platform for the PowerPC processor, and announcing open licensing of the Mac OS.

This article will first look at the hardware reference platform and what it means to you. From there, it will examine the larger issue—licensing the Mac OS—including Apple's goals and how Apple plans to achieve them.

What Is the Hardware Reference Platform?

A big part of Apple's licensing strategy is the hardware reference platform for the PowerPC processor, which is extremely important news

please turn to page 5

AppleDirections

Volume 2, Number 12

Apple Directions, the monthly developer newsletter of Apple Computer, Inc., 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 Apple Developer Periodicals group within Apple's Developer Press.

Editor
Paul Dreyfus (AppleLink: DREYFUS.P)

Technical Editor
Gregg Williams (GREGGW)

Business & Marketing Editor
Kris Newby (NEWBY.K)

Associate Editor
Anne Szabla

Production Editor
Lisa Ferdinandsen (LISAFERD)

Contributors
Peter Bickford, Alex Dosher, Sandee Karr, Gary Little, Kris Newby, Caroline Rose, Jessa Vartanian

Manager, Developer Press
Dennis Matthews

Manager, Apple Developer Periodicals
Greg Joswiak

Production Manager
Diane Wilcox

PrePress/Film
Aptos Post

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

Editor's Note

(More Than) Equal Opportunity in Education

Apple Computer, Inc., owns a 61 percent share of the market, and Macintosh market-share is expected to increase steadily.

Don't you wish that were true? Well, I'm going to grant your wish: It *is* true. Even better, I think there are clear opportunities for software developers to make a huge difference in this market, as I'll get back to after I give you some numbers.

The market in question is the U.S. K–12 educational market for personal computers. Apple *does* own more than 60 percent of that market, according to Quality Education Data (QED), a research firm that gathers data about U.S. educational institutions.

Granted, that large share of the market still includes a great many Apple II systems, which make up about 40 percent of the installed base of computers in U.S. elementary and secondary schools (that is, kindergarten through 12th grade). But the Macintosh share of the market is substantial and growing, as the following data shows:

- Currently, schools use the Macintosh computer more than any other platform, except Apple II systems; 20 percent of U.S. K-12 computers are Macintosh systems, according to Apple internal data. (By comparison, Windows' current share of the market is between 5 and 10 percent, according to various industry estimates.)
- More than 50 percent of unit shipments to the market in 1993 were Macintosh computers, according to QED.
- In 1993, Macintosh shipments grew at one-and-one-half times the overall K-12 market growth rate, and Apple projects 1994 shipment growth to continue at the same rate.
- Based on current growth rates, Apple projects that by 1996, the Macintosh share of the market will be about 40 percent.

This is a large, growing market; Apple estimates that there are currently 1.5 million teachers, 800,000 administrators, and 45 million students in U.S. K–12 schools. (And, remember, each of those students has at least one parent!) Many school districts have a dream, if not a goal, of one day equipping every student with a computer. That's far from true today, but the student-to-computer ratio is dropping quickly: According to Apple data, in 1991 every 18 students had to share a computer, but by 1996 that ratio will decrease to 8 students fore every computer.

Putting some of those numbers together, within two years there will be nearly 6 million computers on students' desks; about 2.5 million of them will be Macintosh systems, able to run your Macintosh products.

• • •

I'm giving you all these numbers because I think more than a few of you will be able to take advantage of this burgeoning market, at the same time helping Apple cement the Macintosh computer's dominance in it. But I'm also encouraging you to get into this market because there's worthwhile work to be done. In short, I think schools need better software, and you're the only people who can create it.

Watching my two children (ages 8 and 6) attend elementary schools that make substantial use of Macintosh computers, I'm convinced that *the* breakthrough educational application has yet to be designed. Both my kids attend computer literate public schools. The student-computer ratio is about 10 to 1 in each school, and the administration emphasizes integrating technology into the schools' curricula.

In each school, kids use computers mostly to become computer literate and to boost their reading, writing, and arithmetic skills. These are certainly valid purposes, but they aren't going to make the personal computer a necessity for every teacher and student.

Teachers are not likely to adopt technology that teaches skills for which there are already sound, nontechnical educational methods in place. Drill-and-practice tools for language and math fall into that category; why would a teacher use technology to teach those subjects when they're already quite happy with the methods they've used for years?

Even though many kids seem innately drawn to the graphic whiz-poppery and ease-

of-use of the Macintosh computer, it's a rare teacher who's going to adopt technology for its own sake; instead, teachers (and students and parents) want tools that help enrich the educational experience—preferably in these days of program cuts, by going beyond what their educational resources currently allow them to do.

Music; art; in-depth scientific, historical, literary, and cultural exploration; field trips to museums—these are but a few of the study areas that are fast disappearing from elementary and secondary schools in the current economic climate. And these are just the areas that ingenious, multimedia computer software can help put back in the classroom at a price far more reasonable than funds for the new teacher or building that can't be added to the budget.

Don't you think computers will become an absolute necessity when they can take students on a tour of the world's museums, help them simulate the next great California earthquake and study its effects, play and show a symphony orchestra concert, or help teach them to draw? These are the kinds of activities that can make school a vital, interesting place for students, and they're the kinds of applications that today's Macintosh technology makes possible.

So get out there and start designing that breakthrough Macintosh educational solution—before somebody else comes along and does it first. A good place to start is with this month's Market Research Monthly on page 18, which provides a thumbnail sketch of the U.S. K–12 Macintosh market and what its customers are looking for.

Paul Dreyfus, Editor

IndustryWatch: News & Perspective

There's Work to Be Done

Prepared by the Apple Directions *staff*

Microsoft/Intuit Spells . . . Opportunity

Microsoft has agreed to purchase Intuit for $1.5 billion, the largest merger yet between two software firms. The acquisition will give Microsoft instant strength in the market for personal finance software, a market currently dominated by Intuit's Quicken product, which reportedly has a 70 percent share. It's expected that Microsoft will use Intuit's expertise and resources to work with financial institutions on providing online financial services for personal computer users.

Implications/Opinions: Everyone (except those who made money on the deal) can take a moment of silence (or angst, inertia, whatever) at this latest signal of the consolidation of the computer software industry and the Microsoft hegemony. But only a moment, because there's work to do. We could sit around feeling victimized, maybe waiting for antitrust lawyers to try and make it easier to compete. Or we could accept what's happening, look for the opportunities implicit in the deal, and go out and provide our own competition.

In our view, the deal points the way to another hot, new area: combining personal computers, online services, and banks to allow people to manage finances from the desktop. According to a report in the *Wall Street Journal*, banks are scurrying around trying to find ways of providing online services to their customers. Those services will require software front-ends, connection services, databases, and so on, and not everyone will want to use software provided by a large, international firm. In particular, small, local developers may have the opportunity to work with banks in their area to develop online financial services for local customers who don't want to shift allegiances to the large financial institutions that are likely to be prominent in this new field.

Back in the 1950s and 1960s, people were afraid that the huge department stores would kill specialized boutiques. Department stores got bigger and bigger, absorbing revenues that smaller shops had previously garnered and putting some of them out of business. But the boutiques came roaring back, jeopardizing the department stores' very existence because people didn't want to buy all of their clothes, cookware, and appliances from the big guys.

As in the consumer goods market, we think that not all personal computer customers will want to buy software from one large company and do all their banking with one institution. We can be sure that desktop financial management is going to get plenty of air play in the coming months, which will boost everyone who currently has a presence in that market. In other words, if you have ambitions in the market, get busy and take advantage of all the attention it's going to get as a result of the Microsoft/Intuit merger.

The same, general message applies no matter what area of the market you're in. If the biggest companies are focusing on a particular product type or customer, you can bet that part of the market is going to get a lot of attention—attention that small, clever, fast-moving companies just might be able to capitalize on. In fact, you might recall a startup company called Intuit with an easy-to-use personal finance software package and a unique ability to listen to customers and provide customer support. . . .

The Sony Magic Link—PDAs in it for the Long Haul

In late September Sony rolled out the Magic Link PIC-1000 communicator, which uses Magic Cap and Telescript technology, offers users access to AT&T PersonaLink services, and ships free with a variety of useful software. Partway between a "palm-top" computer and a Newton-like personal digital assistant (PDA), it employs a touch-sensitive liquid-crystal display and an optional keyboard. The 1.2-pound device measures 5.2 inches by 7.5 inches, stands 1 inch tall and carries a list price

Apple Directions Online—January

The January issue of *Apple Directions* will be available on AppleLink on December 15. To view the January issue of *Apple Directions* online, follow the AppleLink path Developer Support:Developer Services:Periodicals:Apple Directions:Apple Directions January 1994.

of $999. In announcing the product, Sony downplayed the possibility of immediate success, suggesting that the first Magic Link is only an effort to learn more about the market for personal communicators.

Implications/Opinions: Newton developers, take heed. Sony's introduction of the Magic Link is a vote of confidence for the entire PDA concept. Sony wants to find out how people use communications assistants, who those people are, and what they'll be looking for in future PDA devices; they'll also be exploring how to build marketing and distribution efforts. Although the market for PDAs is small—the *Wall Street Journal* estimates that the current market for pen-based computing devices is less than 200,000—Sony is joining a host of others in expecting the market to grow to enormous proportions. The new device's success will be gauged less by unit sales than by the value of the lessons learned.

Sony's entry into the PDA/personal communicator product area only gives the entire category more attention, potentially speeding up the growth of the market and widening the audience for the PDA products you've already developed. If you've already got a product on the market, you may not be experiencing huge sales, but you can pride yourself on having been there first. Later, what you learn today will likely turn into tomorrow's profits.

New Multiprocessing Standard to Compete With Intel

A group of companies that includes major personal computer (PC) heavyweights, but excludes Intel, is working together to create a standard for PC multiprocessing. The new standard, SLiC/MP, is backed by microprocessor vendors Advanced Micro Devices (AMD), Cyrix, IBM, and Motorola; chipset vendor Opti Inc.; and Microsoft. This standard is expected to compete with Intel's recently announced Advanced Processor Interrupt Controller (APIC).

The SLiC/MP standard will be used in systems based on 80x86-compatible chips from AMD and Cyrix and in PowerPC processor– based systems from Motorola. It is currently limited to two processors at once, but will be extended to support more. Intel's APIC already supports up to eight processors.

Implications/Opinions: IBM's control of the PC market was weakened by competitors promoting other standards such as SuperVGA graphics and the VESA local bus. Similarly, this promotion of a competing multiprocessing standard could be yet another event step in weakening Intel's leadership of the 80x86-compatible market.

Doom Is Something When You Give It Away

Doom II: Hell on Earth, the sequel to the violent PC-DOS game, *Doom,* was released last month in a New York City disco amid tremendous fanfare. Its distributor, GT Interactive Software, anticipates that the game will sell over 1 million copies by January; dealers have already ordered more than 500,000 copies of the game.

Implications/Opinions: Why would we report the release of a PC game in *Apple Directions*? Because of the lesson implicit in the initial success of *Doom.* You might recall that a part of the original game, developed by id Software, was initially distributed for free over the Internet. Millions of copies were downloaded; many players were immediately hooked, and more than 150,000 of them purchased the entire game. Other vendors, recognizing the success of id's "the first-one-is-free" strategy, are following suit with their newest games.

To be successful, your product distribution plans must be increasingly creative. Software dealers' shelves are only so large, and it's up to you to figure out how to get your product into your ultimate customers' hands—even if it means giving it away. One emerging distribution channel is the Internet. As the Editor's Note pointed out last month, it's expected that the Internet will soon be a significant purchasing medium for consumers.

Voluntary Ratings System for Entertainment Software

The Recreational Software Advisory Council (RSAC) was recently formed by the Software Publishers Association (SPA) to provide a voluntary rating system for entertainment software. SPA formed RSAC in answer to congressional calls for the software industry to police itself or be regulated, and to allay the fears of parents over the spread of violent games (like the aforementioned *Doom*) and software with sexual content.

The system allows software firms to indicate the appropriateness of games, multimedia titles, and other recreational software for use by children. Software is rated according to its language, use of violence, and sexual content. To determine the rating, the firm publishing the software fills out a questionnaire that's part of a special application provided by RSAC. The application then provides the rating, which is approved by the executive director of the council. It's expected that the first software packages displaying the new ratings will be on shelves for the 1994 holiday buying season.

Implications/Opinions: The ratings system strikes us as a sound way to give parents more information to help them and their children figure out what software is appropriate for them. Better that than to have Congress impose a ratings system, or to go the way of the electronic games industry, which uses a three-member board made up of employees of the top games companies (such as Sega and Nintendo) to rate products—a system that could be more arbitrary than the RSAC's computerized ratings.

If you're developing home entertainment products, you may want to find out more about the system. If the system catches on as it's supposed to, parents probably won't buy your product if it doesn't display a rating. For information, contact Glenn Ochsenreiter, Acting Executive Director, Recreational Software Advisory Council, 1718 M Street NW, Suite 139, Washington, DC 20036; phone: 202-293-3055; fax: 202-785-3197; Internet: glenno@cap.gwu.edu; CompuServe: 76004,125.

Desperate Times, Desperate Measures

A British software industry trade group is offering bounties of as much as £2,500—nearly $4,000—to people informing on British firms that use pirated software. The Business Software Alliance (BSA) announced this somewhat unusual measure to stem software piracy in Great Britain, where it's estimated that more than half the software in use is unauthorized.

Implications/Opinions: Piracy continues to cost the software industry billions of dollars a year. In some markets, it's estimated that well over 90 percent of software is used in violation of copyright laws. You need to think about how to protect your software from piracy, because it's highly unlikely that efforts like the BSA's will have much of an effect, and even less likely that organizations in the rest of the world will take similar action. For some ideas, see "Antipiracy Technologies: Some Modest Proposals" in the June 1992 issue of *Apple Direct.* (You can find it on the Reference Library Edition of the Developer CD series, path—Reference Library:Periodical:Apple Directions:Apple Direct 1992.) ♣

Strategy Mosaic

This Changes Everything

continued from page 1

for developers. The news story on page 1 provides details, but here's a brief summary:

- This hardware reference platform is a PowerPC hardware architecture that is being designed to run multiple operating systems, such as the Mac OS, AIX (IBM's industry-standard implementation of UNIX®), OS/2 for the PowerPC, and Windows NT.
- Properly written Power Macintosh software will run on hardware reference platform computers from any vendor.
- The hardware reference platform is an open architecture that manufacturers can build using standard components.
- Apple, IBM, and Motorola are participating in this hardware reference platform standard and will jointly contribute to its development.
- Delivery of the first hardware reference platform computers is expected in 1996.

Significance to Developers

Whether you develop hardware or software, this hardware reference platform is extremely important news for you. You can now commit to supporting the Power Macintosh architecture (that is, the Mac OS running on the PowerPC processor), knowing that it will become a higher-volume platform in the future. The fact that Apple, IBM, Motorola, and other hardware vendors will be supporting a single hardware architecture will increase the market for "native" software and add-on hardware products.

Apple, IBM, and Motorola are very interested in selling as many hardware reference platform computers as possible, and any Power Macintosh software that you develop now can sell to a much larger audience in the future. That's because, when running the Mac OS, every hardware reference platform computer will be a complete Power Macintosh system. And, as the number of hardware reference platform computers increases, so does the potential for you to profit from supporting the Mac OS.

Significance to Customers

This hardware reference platform is good news for customers because it gives them choices and security. They'll be able to buy hardware reference platform computers from a variety of vendors (an important point for many business buyers), and they'll be able to run their choice of major operating systems. Because of widespread acceptance of this hardware reference platform, customers will also have a very large selection of software from which to choose. A wide selection of hardware vendors, operating systems, and applications will give customers a sense of security—peace of mind that, whatever their future needs are, they will be able to find the software they need to get their work done.

Significance to Apple

On the hardware side, this hardware reference platform will allow Apple to bring the benefits of Power Macintosh next-generation computing to a wider audience.

On the software side, this hardware reference platform will greatly strengthen the position of the Mac OS. The Mac OS has two important advantages over all the other operating systems that this hardware reference platform will support. First, it is currently the only mainstream operating system for the PowerPC processor, as evidenced by the fact that the Power Macintosh platform currently outsells all other RISC-based computers combined.

Second, the Mac OS is the only RISC operating system with a significant installed base of applications (344 native Power Macintosh applications, as of the end of October)—and it will continue to hold this lead over competing operating systems as time goes by. This, combined with Apple's plans to license the Mac OS to other vendors, will make the Mac OS the most popular operating system for hardware reference platform computers, thereby encouraging developer adoption of the Power Macintosh platform.

Significance for the Alliance Partners and the Industry

This hardware reference platform will make the PowerPC platform the computing platform of choice for RISC and will clearly position it as an open alternative to today's Windows/Intel platform. It will also encourage more hardware and software vendors to become early adopters of PowerPC technology.

A processor does not a standard make—and, so far, there has been some uncertainty over the compatibility of computers designed using the PowerPC processor. This hardware reference platform sends a clear signal to customers, developers, and the computer industry: this hardware reference platform will be the most important cross-vendor, multiple-OS hardware platform for the PowerPC processor.

Licensing—The Bigger Picture

In addition to its traditional strengths in markets like publishing, education, and multimedia, Apple has, in the past few years, been growing at or slightly above the rate of the rest of the personal-computer industry. But Apple's strategy is to gain market share from competing platforms. To do that, Apple must sell more Mac OS computers than projected by Apple's current growth rate—and licensing is the best way to achieve the level of growth needed to gain market share. Mac OS clones (expected sometime in 1995) are one form of licensing, as are the hardware reference platform computers (due in 1996). In fact, Apple expects the installed base of Mac OS–capable computers to increase significantly soon after hardware reference platform computers become available. (See the figure "Licensing will help Apple gain market share" on page 6.)

Apple's Plan for Licensing

Apple's plan for licensing includes affiliating itself with other companies in ways that allow Apple to increase its market share by using the other companies' balance sheets. For example, take a hypothetical country X, in which Apple has virtually no presence. Suppose that company ABC has a strong PC-compatible presence in country X. If company ABC licenses the Mac OS, it will then commit both money and resources (strong distribution channels, for example) to sell its Mac OS computers. Not only does Apple get the use of ABC's money, it also benefits from ABC's strengths in country X. And since Apple has a limited presence in country X, it sells more Mac OS computers in that country but does not lose significant sales to channel conflict (commonly called "cannibalization").

Apple's licensing does not limit ABC's sales of Mac OS computers to any geographic area—ABC can sell its Mac OS computers in countries X, Y, and Z, if it desires. However, if ABC does not have the same market advantages in countries Y and Z as it does in X, then it may well decide not to market Mac OS computers in those countries.

Apple will begin with licensees that have been carefully chosen to maximize the potential for both companies' success, then move as quickly as possible to open licensing. However, licensing is a very complicated undertaking, both technically and legally, and it will take time for Apple to create the necessary infrastructure and processes. The support required includes, but is not limited to

- technical support
- engineering support
- technical documentation
- marketing support
- temporary startup support

to new licensees

What Apple Is Licensing

One question that people will be asking is "What, exactly, is Apple licensing?" Actually, Apple is pursuing several forms of licensing. Some licensees will essentially be repackaging existing Macintosh models. Other licensees will buy the necessary components and technologies from Apple to create their Mac OS computers. Obviously, the simplest forms of licensing will be the first to market, but by the time hardware reference platform computers become available, Mac OS computers of various lineages will be available.

How Apple Will Thrive

The *big* question that people will be asking is "Once Apple moves to open licensing, how will Apple make its money?" Apple believes that the reputation of the Mac OS and Apple's key role in this jointly defined hardware reference platform standard will earn Apple a major share of the hardware reference platform pie.

When hardware reference platform computers become available, their vendors will be spending much more on marketing than Apple can alone. Not only will this result in significantly higher visibility for the Mac OS platform, it will associate the Apple name with this joint hardware reference platform standard and will influence many buyers to purchase Apple-labeled hardware reference platform computers.

Apple believes that this and other factors will significantly outweigh the potential for channel conflict. The result will be that Apple will do well for its developers, make the transition to open licensing, and become a more profitable company in the process.

Why Commit to Mac OS?

Apple is pursuing several strategies to gain market share, which include licensing the Mac OS to other vendors and participating in the hardware reference platform standard with IBM and Motorola. This hardware reference platform is particularly important because it accelerates the PowerPC processor's growth as the first RISC computer platform with broad appeal.

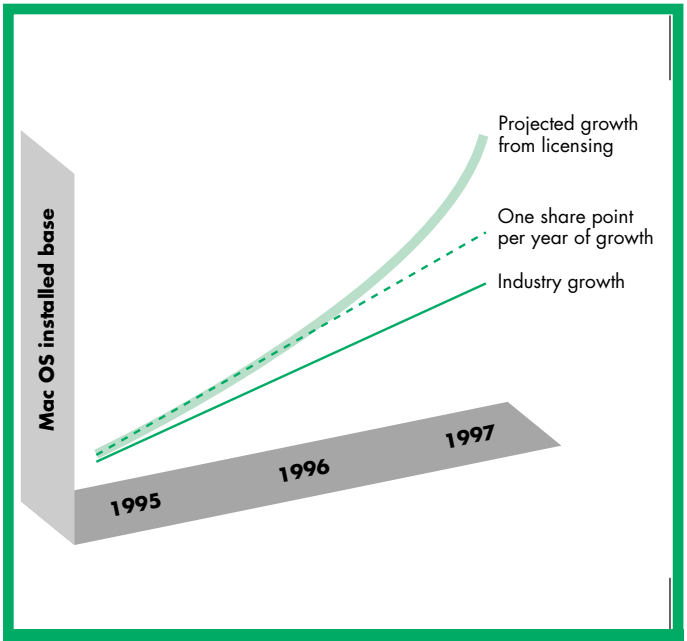
What does this mean to you? Quite simply, more sales of your products, and the assurance that the Mac OS will become more important as time goes by. The most important thing for you to remember after reading this article is that keeping your Power Macintosh applications current with the Mac OS is the best way to prepare for the hardware reference platform computers.

When you reevaluate your company's commitments to the Mac OS and Windows platforms, you should consider the futures of both. Because of the increased resources that Windows 95 needs, the changes it will require to Windows applications, and customer resistance to upgrading their systems, Windows 95 (formerly known as "Chicago") is not at all certain to become as widely adopted as Windows 3.1 is. In fact, if Windows 95 adoption is light, Windows 3.1 may become a "frozen" standard that Microsoft can no longer manipulate to its advantage. (In the early 1980s, the IBM AT hardware design became such a standard, severely eroding IBM's domination of the emerging PC-compatible market.) With IBM's OS/2 operating system and an increasing number of Macintosh computers able to run Windows 3.1 software, developers—and users—may be content to stay with Windows 3.1.

Also, the 80x86 architecture has only so much life in it, and neither Windows 3.1 nor Windows 95 will make the transition to a RISC processor—both are too closely tied to the 80x86 instruction set. (Windows NT, however, can run on top of different processors—including the PowerPC processor used in all the hardware reference platform computers.) Intel has announced its intent to move to some kind of RISC architecture by the end of the decade, so the future of 80x86-based Windows software is uncertain.

The prospects for the Mac OS are considerably brighter. The PowerPC processor is a much newer, more powerful design, and the Mac OS gives you a way to deliver that power to an existing, well-established market. That market will grow as other companies start selling Mac OS computers. And the next major revision of the Mac OS, Copland, will strengthen the platform with fundamental enhancements to the core operating system and the user interface, as well as numerous improvements in other areas.

Finally, don't underestimate the importance of the Mac OS to your worldwide business opportunities. Windows is not yet a standard in the fastest-growing markets outside the United States, and the hardware reference platform represents a real option for the majority of businesses in countries such as Japan, where DOS is still dominant. Through WorldScript technology, Mac OS–based software can be more easily localized into non-Roman languages, making it easier for your products to get an



Licensing will help Apple gain market share. Licensing, including Mac OS-compatible computers in 1995 and the hardware reference platform–compatible computers in the second half of 1996, will allow the Mac OS to gain additional market share. This graph is meant to illustrate general trends but not specific projections.

early advantage in the fast-growing emerging markets of the Far East.

You've already invested in the Mac OS platform, and the good news is that in the future, you'll get a significantly better return on the investment you've already made. Now is the time to increase your commitment to the Mac OS platform and "lock in" future profitability. Apple hopes that the announcement of its licensing plans, and the Apple/IBM/ Motorola hardware reference platform standard, will persuade you to do more with the Mac OS. The future of this innovative platform has never been brighter. ♣

Apple News

New Hardware Reference Platform

continued from page 1

specifications will run largely unmodified on the new hardware reference platform, the new platform will combine many technical characteristics and features from Apple's Power Macintosh architecture and from the current IBM and Motorola PowerPC Reference Platform, which a number of hardware and software vendors have already adopted.

Each company intends to take technical and financial responsibility for porting an operating system to the common platform. Apple will be responsible for porting the Mac OS, IBM will port AIX and OS/2 for the PowerPC, and Motorola will port Windows NT. The three companies will continue to work together with other vendors on porting their operating systems to the new platform.

In a joint statement, the three companies said, "The PowerPC microprocessor has already established itself as the industry's price/performance and technology leader. By combining that technology with an open hardware platform and the industry's broadest range of operating system availability, we are creating a new industry standard—one whose entrance means greater opportunities for developers, greater performance for customers, and greater competition for the industry."

This new hardware reference platform aims to create new business opportunities and alternatives for software developers and hardware vendors who demand a higher return on their investment. It will also allow vendors to invest once in developing a product and have it reach a huge installed base of computers from a variety of vendors.

The objective is to achieve 100-percent compatibility with current PowerPC processor–based hardware and software products, while creating a new standard that offers future capability for advanced features. These features include designs for the high-performance peripheral component interconnect (PCI) bus, system startup that works independently of the operating systems, and industry-standard components and I/O systems for compatibility.

More technical information on the new hardware platform will be published in spring of 1995. The alliance members plan to present prototypes of the systems based on the specification in 1995, and the first computers are expected to be available in 1996.

PowerPC is a family of RISC-based processors developed jointly by Apple, IBM, and Motorola, and a key plank in the 1991 technology alliance between the three companies. Since then, the companies have garnered broad industry support for the PowerPC processor family. Currently, over 25 companies are developing or marketing PowerPC systems, including Apple, Bull Systems, Canon, FirePOWER, Hitachi, IBM, Motorola, 3DO, the Taiwan New PC Consortium, and Toshiba.

New Power Macintosh System: Industry's Fastest PC

Apple Computer, Inc., announced a computer in November that further extends the Power Macintosh performance leadership over systems based on Intel's Pentium microprocessor. The Power Macintosh 8100/110 computer, which runs at 110 megahertz, is the most powerful personal computer on the market.

The newest computer from Apple joins the current Power Macintosh product line based on the PowerPC 601 RISC microprocessor—the Power Macintosh 6100/60, 7100/66 and 8100/80 computers—which is providing an ever-widening base of customers for your native Power Macintosh products. Apple has already sold more than 600,000 Power Macintosh systems, and expects to have sold 1 million by March 1995.

Apple expects the Power Macintosh 8100/110 to appeal to power-hungry customers such as professional publishers who routinely work with massive image files and computation-intensive design, graphics, and imaging applications. The new system is also expected to appeal to customers in the multimedia and engineering markets.

The Power Macintosh 8100/110 computer ships installed with 16 megabytes of memory (expandable to 264 MB), a 2-gigabyte hard disk, a double-speed CD-ROM drive, an Apple SuperDrive 1.4 MB floppy disk drive, and on-board Ethernet. It's driven by a 110-MHz version of the PowerPC 601 microprocessor, and includes an integral floating-point processor, a 32-kilobyte internal cache, and 256K level-2 cache. It can support a wide range of peripherals with its nine built-in ports and includes three NuBus™ slots for expansion cards.

Like all Power Macintosh models, the Power Macintosh 8100/100 computer runs virtually all 680x0-based Macintosh applications in emulator mode at 68040 speeds; "native" applications accelerated for PowerPC processor–based systems run two to six times faster than 680x0 applications on the most powerful 68040-based Macintosh Quadra systems. Additionally, the Power Macintosh 8100/110 computer can run DOS and Windows applications using SoftWindows™ from Insignia Solutions.

With a U.S. Apple price of $6,379, the Power Macintosh 8100/110 is available today in limited quantities in the United States and selected regions worldwide. Volume shipments of the product are expected to start in December 1994.

Macintosh Performa 6100: PowerPC Hits the Consumer Market

Extending its price and performance leadership into the

consumer market, Apple Computer, Inc., expanded its Macintosh Performa line of computers last month with five new computers—the Macintosh Performa 6100 series—featuring the RISC-based PowerPC 601 microprocessor. Macintosh Performa systems, marketed specifically to home customers, are sold in thousands of consumer electronic stores, department stores, and shopping clubs worldwide.

If you're one of the hundreds of developers who've already released software that runs in native PowerPC mode, you'll now have the opportunity to sell that software to customers in the home market. If you haven't taken your consumer software products "native" yet, you're going to want to think about doing so soon. While the new systems will run 680x0 software in emulation mode, once consumers have access to the increased power of RISC processing, they're going to want software that takes advantage of its two- to four-times performance increase over 68040-based systems.

Macintosh Performa 6100 series systems are based on the PowerPC 601 microprocessor running at 60 megahertz. Macintosh Performa 6100 series computers ship with 8 MB of memory and either a 250 MB, 350 MB, or 500 MB hard drive, depending on the model. Each system is equipped with Apple's internal double-speed, tray-loading CD-ROM drive, the Apple Multiple Scan 15" Display with built-in stereo speakers, an external 14.4K BPS Global Village fax/modem (send/receive fax), and the AppleDesign Keyboard.

On the software side, the Macintosh Performa 6100 series computers ship with System 7.5 and a variety of application software and CD-ROM titles specially chosen to appeal to home users. With Insignia Solutions SoftWindows, Macintosh Performa 6100 computers can also run most DOS and Windows applications.

For now, Macintosh Performa 6100 series computers will be available only in the United States, although Apple expects to release PowerPC processor–based Performa systems in other countries soon. Prices for the Performa 6100 series computers range from $2,600 to $2850.

Mac OS SDK Includes Redistribution Rights for Most System Extensions

The just-released Mac OS Software Developer's Kit (SDK) just became even more efficient. In addition to combining programming information for more than 30 extensions to the Macintosh operating system in one package, the Mac OS SDK also gives you rights to redistribute many of the extensions it contains.

You may incorporate the software into your own programs and distribute it (in object code form only) to your customers according to the terms of the Mac OS SDK Software License. (Previously, you had to pay licensing fees to include the separate extensions with your products.)

Not all of the extensions can be redistributed free of charge, however; some must still be licensed specially from Apple, while others require you to include specific language on your product's package. To find out exactly which software you can redistribute without additional charge or effort, see the Mac OS Software License, which is included in printed form with the Mac OS SDK CD-ROM, and the redistribution information on the CD itself.

Here's a partial list of the software on Release 1 of the Mac OS SDK that you can redistribute:

- AppleScript run-time files
- XTND run-time files
- MacODBC run-time files
- Serial Switch
- Thread Manager
- Sound and Sound Manager
- File System Manager
- Macintosh Drag and Drop
- AppleSearch Client, AppleSearch Authentication, AppleSearch Communication, and ASClientLib.o
- Apple Guide
- ColorSync
- Apple MIDI Driver and MIDI Manager
- Telephone Manager and TelMgrGlue.o

For a complete list of software on the SDK, contact APDA (see page 24). Also, last month's *Apple Directions* included a partial list of the first Mac OS SDK's contents (see page 11, "New Mac OS SDK Saves You Time and Money").

One-year subscriptions to the Mac OS Software Developer's Kit, which is shipped quarterly on CD-ROM, are available from APDA. One subscription is included in Apple's monthly mailing to members of the Apple Partners program worldwide.

ATG Announces Awards for Internet Programming Efforts

Last month, the Advanced Technology Group (ATG) of Apple Computer, Inc., rewarded 11 developers for contributing a wide variety of tools to help Macintosh users navigate the Internet. Recipients of the awards—called the "Cool Tools" awards—received an Apple Power Macintosh 7100 computer.

The Internet, which was until fairly recently available only to researchers and scientists, has evolved during the past several years into a worldwide network connecting more than 75 countries and millions of users in small businesses, libraries, schools, homes, and government offices.

"Access to the Internet has been made simpler by the efforts of many programmers—many of them working with little monetary reward—to design and distribute a wide variety of tools to navigate the millions of files and thousands of networks that comprise the Internet," said Rick LeFaivre, vice president of ATG. "Through the creation of these awards, our goal was to recognize the work of some of these unsung heroes who have made very significant contributions in making it easier to navigate the Internet.

"Consistent with the charter of Apple's Advanced Technology Group to develop new hardware and software technologies, we hope this recognition encourages the continued development and evolution of the wonderful resources available on the Internet," LeFaivre added.

It's unclear whether similar awards will be made in the future, although ATG will continue to provide support for developers working on innovative software that makes it easier for Macintosh customers to use the Internet. If your work falls into that category, you can contact Steve Cisler at Apple (AppleLink: CISLER1; Internet: sac@apple.com).

The following are the Macintosh Cool Tools Internet Award recipients and their projects. All of the award-winning programs started off as free software or shareware; some eventually became commercial products. Recipients were

selected by a panel of Apple employees.

• The Internet Society, Reston, Virginia, for its efforts to foster a global environment conducive to the easy exchange of information and the rapid development of standards and new software (phone: 703-648-9888; fax: 703-648-9887: Internet: amr@isoc.org).

• Steve Dorner of QUALCOMM Incorporated in San Diego, California, for Eudora, an electronic mail client for Internet users (phone: 800-2-EUDORA; Internet: sdorner@qualcomm. com).

• Chuck Shotton, Houston, Texas, for MacHTTP, a World Wide Web server for the Macintosh computer (phone: 713-794-5650; Internet: shotton@oac.hsc.uth. tmc.edu).

• Peter Lewis, Perth, Australia, for FTPd, an anonymous file transfer server, and Anarchie, an FTP client to search for and retrieve public files on the Internet (Internet: peter.lewis@info. curtin.au.edu).

• Weather Underground, University of Michigan, for BlueSkies, a gopher client for browsing, viewing, and reporting real-time weather and environmental information in an interactive graphic and text format (phone: 313-936-0491; Internet: blueskies@umich.edu).

• John Hardin of EINet, Austin, Texas, for MacWeb, a hypermedia World Wide Web client for the Macintosh computer (phone: 800-844- 4638; Internet: macweb@einet.net).

• National Center for Supercomputer Applications, Urbana, Illinois, for Mosaic for the Macintosh computer, the cross-over application that has helped to spur interest in the Internet for both commercial and noncommerical users (phone: 217-244-3473; Internet: mosaic-mac@ncsa.uiuc.edu).

• Aaron Giles of Cornell University Medical College, New York, for JPEGView, a graphic utility that allows the user to view compressed images on the World Wide Web or Gopher or those retrieved from anonymous FTP servers on the net (phone: 212-410-2781; Internet: giles@med. cornell.edu).

• John Norstad of Northwestern University, Evanston, Illinois, for Newswatcher, a Usenet news reader (Internet: j-norstad@nwu. edu).

• Cornell University, Ithaca, New York, for CU-SeeMe, a conferencing tool used by elementary schools, individuals, and other organizations around the world for low-cost video communications (phone: 607-255-7566; Internet: r.cogger@cornell. edu).

• University of Minnesota, Minneapolis, for the TurboGopher client and GopherSurfer server (phone: 612-625-1300; fax: 612-625-6817). ♣

Technology

Inside This Section

develop Issue 20: The Answers Are Here

Do you wonder what AOCE catalogs are useful for, beyond supplying e-mail addresses or browsing network devices? Do you know how to manipulate the Finder from your application? Do you know about the PowerPC version of NetWare, the networking standard in the DOS/Windows world? Why stay in the dark when you can learn about all this, and much more, in Issue 20 of *develop,* Apple's award-winning technical journal? Its enlightening articles include these:

• "Getting the Most out of AOCE Catalog Records" introduces you to writing AOCE templates, which let you extend catalogs to store new kinds of data. It provides an example template that extends the built-in User record.

• "Exploring Advanced AOCE Templates Through Celestial Mechanics" shows how to use some of the more advanced features of the template mechanism, to specify how the Finder displays your data and how the user interacts with it.

• "Scripting the Finder From Your Application" tells how the OSL-compliant System 7.5 Finder opens the door to full scriptability and to direct manipulation through Apple events. You'll learn how to talk to the Scriptable Finder from your application.

• "NetWare Development on PowerPC" is an introduction to the NetWare environment that includes a simple sample module. With NetWare now on the Power Macintosh, you may want to write your network products to use it.

• "Make Your Own Sound Components" gets you started with plug-in components

please turn to page 17

CD Highlights

Reference Library Edition, December 1994

New this month is the Master Subject Collections folder within the Subject Index folder, which contains a collection for each New Inside Macintosh subject area (as well as a few others). Each collection includes references to *Inside Macintosh,* Macintosh Technical Notes, relevant *develop* magazine articles, and other documents in DocViewer format. These collections should be your first stop when searching the available literature for answers to your technical questions.

Another big improvement this month is in the Macintosh Technical Notes. Thanks to Alex Solinski, summer intern Dan Sletten, and the Whole Sick Crew in Apple DTS, obsolete notes have mostly been removed, the remaining notes rebuilt as separate DocViewer files to simplify future updating. See the Document Collections folder within the Subject Index folder for Technote collections by subject.

And now, the usual Big Pile o' New Stuff.

Reference Library Edition

ABS Technical Notes

ABS Technical Notes contain the latest bugs, tips and tricks for developers of software for AppleSearch, AppleShare, AppleTalk Remote Access, Apple Workgroup Servers, DAL, IP Gateway, and SNA•ps.

develop

This is the electronic version of *develop* Magazine, Issue 20. For more details about the latest issue of *develop,* see the article on this page.

Developer Notes

Included here, along with our regular archive, are developer notes for two new products: the Power Macintosh 8100/110 and the DOS Compatibility Card.

The Power Macintosh 8100 features increased processor speed (to 110 MHz) and adds a new 1 MB cache SIMM. The DOS Compatibility Card features a Dual Processor implementation based on PC AT and Macintosh core architecture that shares the Mac's I/O hardware and provides DOS compatibility.

The Display Device Driver Guide describes how Display Manager 1.0 communicates with display devices. It provides the details necessary for programming display device drivers that fully support the Display Manager features on the Macintosh platform.

Developer University Course Info

This folder provides information on the types, cost, and location of training available to Apple developers through Apple's Developer

University. It helps customers determine which training will be appropriate to their needs, and includes current course descriptions, class dates and locations, and information on how and where to get self-paced materials. (The Developer University schedule can also be found on page 17 of this issue.)

Inside Macintosh—Advanced Color Imaging (Draft)

Inside Macintosh is a collection of books, organized by topic, that describe the system software of Macintosh computers. Together, these books provide the essential reference for programmers, designers, and engineers creating applications for the Macintosh family of computers.

This electronic book contains preliminary drafts of chapters describing the following collections of system software routines:

- the Color Manager
- the Color Picker Manager, version 2.0
- the ColorSync Utilities, versions 1.0 and 2.0
- the Display Manager, version 1.0
- the Palette Manager

The chapters in this book describe how to use these managers to enhance your application's color capabilities. To implement core graphics capabilities, your application should use QuickDraw or QuickDraw GX. The book *Inside Macintosh: Imaging With QuickDraw* describes how your application can use QuickDraw to create and display Macintosh graphics, and how to use the Printing Manager to print the images created with QuickDraw. The *Inside Macintosh: QuickDraw GX* suite of books describes the QuickDraw GX object-based graphics programming environment for creating, displaying, and printing graphics.

The Color Picker Manager version 2.0, the ColorSync Utilities version 1.0, and the Display Manager define new routines that have never been documented in *Inside Macintosh* before (version 2.0 of the ColorSync Utilities is currently under development). The chapters for these managers represent the most complete technical reference currently available, but late changes may not be reflected in the documentation.

RAMDisk 1.1

This folder contains a sample RAM disk system extension written in C that can be compiled using MPW, Symantec, or Metrowerks compilers. This release fixes a bug in GrowUnitTable (in RamINIT.c), which broke expansion of the Unit Table. This sample code shows how to integrate files of type 'INIT', 'cdev', and 'DRVR' in one package. It also shows how to generate code using multiple development environments, and demonstrates some proper techniques for driver installation.

Snippets Update

This folder contains a new batch of code snippets, which will be rolled into the Snippets folder on the next Tool Chest Edition (February 1995).

- C PACKman: This snippet shows how to implement a simple Chooser Package, as described in *Inside Macintosh: Devices;* it's updated from Scott "Zz" Zimmerman's PACKman Pascal sample. It includes the Think C version; only minimal effort is required to generate a version usable with MPW or Metrowerks compilers.
- Color Marquee: This snippet shows one technique of implementing a colored "marching ants" marquee, which is often used for marking the area over which the mouse is dragged. This code allows you to open a PICT file and drag to select part of the picture, and it illustrates how to provide user interface feedback based on such selections.
- ColorizePict: This snippet illustrates how to use the CopyBits routine to "colorize" a picture. It uses the standard Color Picker to select a foreground and background color, and CopyBits colorizes the picture.
- CustIcon: This snippet shows how to add a custom icon to a document from your application. It uses some interesting features of the icon utilities package (documented in *Inside Macintosh: More Macintosh Toolbox*). When you choose Get Info for a document, the icon that the Finder displays in the top-left corner of the Info window can be replaced with a custom icon.
- Dialog popups: This snippet demonstrates the use of the System 7 pop-up control cdef in a program using modal dialog boxes; it also works around a problem with the System 7 Dialog Manager routines, whereby the standard filter procedure reenables a disabled button.
- Icon play: This snippet demonstrates the use of the System 7 icon utilities and the System 7 pop-up control cdef in a program using modal dialog boxes; it also shows how to use the System 7 pop-up menu, how to plot icons from icon families, and how to put 'SICN' resources into pop-up menus.
- MCActionProc: This snippet demonstrates the use of an action filter function for QuickTime movies. This filter function resizes the window whenever the user hides the controller, and handles mouse-down events in the content region of the movie window, allowing a user to navigate through the movie by dragging left and right. The code also works around a QuickTime 2.0 problem with the Drag Manager.
- Movie SetTrackGWorld: This snippet illustrates how to draw over a frame of a QuickTime movie as the movie is playing. The technique consists of drawing a movie into a graphics world that you create, and calling the transfer procedure after each frame is drawn; you draw over the movie frame, and then draw the composite frame to the screen.

The code draws a frame marker on each new frame as the movie is displayed. These markers come from a marker picture. Each marker is 32 x 32 pixels; the picture contains ten of them, and the appropriate marker is copied from the picture to the movie frame.

This code is not intended as a complete example. Because it clips markers to a circular shape, and uses CopyBits without trying to optimize the drawing, the technique is slow: it displays only about eight frames per second (fps). With optimization, the QuickTime team has seen rates of up to 30 fps. To optimize the CopyBits calls, read the technical note "Of Time and Space and CopyBits," and use rectangular clipping regions.

- PictInfoTest: This snippet shows how to use the Picture Utilities to sample an image, and covers basic graphics world techniques for offscreen drawing and manipulation of images. It displays PICT files into a buffered window, and uses the Picture Utilities to sample the colors in the image. It provides a simple technique to use when true color fidelity is not absolutely essential.
- ScreenDump: This snippet shows how to dump an area of the screen to a window.
- Simple Imagecompressor: This is a simple image compression program that shows how to use the Image Compression

please turn to page 17

Human Interface

Throwing Away Your Best Ideas

By Peter Bickford

Every once in awhile, somebody asks me which tools they should use for prototyping user interfaces. Personally, I tend to favor tools like HyperCard or Macromind Director, but I'm always keen to find any sort of prototyping tool that will help me work better.

Recently, I took the opportunity to see what other sorts of tools were being used by more technical folks. After observing what seems like dozens of programmers, I've come to a rather startling conclusion: Most programmers don't prototype in C, Pascal, HyperCard, or anything else. They prototype in concrete. Other popular prototyping materials seem to include granite, cast iron, and a new one: that sort of "ruggedized" trash-can plastic that you can kick around all day but that eventually bounces back to its original form.

Strangely, the prototypes may look like mere screen shots or design sketches, but their true nature is revealed when you try to get the programmer to make changes. That's when you find out that although you can paint it and pretty it up a bit, the design was set in stone from the very start.

Call me old school if you like, but this isn't what I thought prototyping was all about.

Rapid Prototyping vs. Quick-Dry Mental Cement

The basic idea behind building prototypes is that design ideas can be improved upon by showing them to others in a physical form. If you've ever seen the glazed look in users' eyes when you describe the data flow diagrams behind a big business system, you know how hard it is to get meaningful feedback from abstract sketches. Mock up the screens they will be using, on the other hand, and you'll get comments galore.

Rapid prototyping is the practice of whipping prototypes up quickly, testing them out on others, then tossing the old model out and redesigning based on the feedback you've gotten. If you're doing it right, you can expect the final product to be quite different than the first prototype. This is called the "design progression," or "progress" for short.

If this process is going to work, however, you need to master a certain cruelty toward your own designs. Writers call it "murdering your children": being willing to kill your best ideas if they just aren't working out. Unless you're willing to be wrong on your first try, you'll never give yourself a chance to start over and really get it right later on.

Egoless Programming and the Value of Mistakes

There was an interesting book by Gerald Weinberg published several years back called *The Psychology of Computer Programming.* Weinberg wrote it to explain what drives us on to weather such job hazards as 15-hour searches for your program's elusive memory leak.

I believe Weinberg nailed it when he talked about the incredibly rewarding sense of creation that comes with taking a program concept and giving it form, seemingly through sheer force of will. In this way, being a programmer is like being an artist, with two major exceptions. The first is that artists tend to have better fashion sense than we do. The second is that for at least part of task, there are standards for evaluating our work.

Perhaps it's this mixture of art and engineering that makes programming such a highly personal endeavor. We put more of ourselves into our work than we tend to admit. Unfortunately, while artists are given great latitude for stylistic differences, in our profession we tend to think that our ideas are either right or wrong. Being wrong hurts, and—the human psyche being what it is—it's not surprising that we try to avoid being wrong as often as possible.

But being wrong from time to time is one of the best ways to learn. Refusing to be wrong is a perfect recipe for stagnation (as well as a habit that makes you bad company at parties). Weinberg addresses this by calling for the practice of "egoless programming." The idea is that mistakes are something to be laughed at, even prized, since they point out how much you are learning. Instead of valuing absolutely infallible programmers, Weinberg advocates giving honors to the programmer who can claim the most outrageous mistakes.

Now, I'll grant you that it may be a bit much to loudly laugh, "Geez, I must be the *worst* programmer in history! I spent—get this—six and a half hours debugging before I realized I had the wrong include file!" Still, Weinberg's ideas do have merit. Nor is he alone in advocating his ideas. Pick up just about any business book and you'll see mention of "learning organizations" and "the value of mistakes." My favorite is the story of how IBM's venerable chief had to confront a senior executive whose failed idea had cost the company millions. The downcast executive came into Akers's office and offered his resignation. Instead, Akers is said to have told him something like, "Are you kidding? I can't let you go—I just spent millions of dollars educating you!"

It May Be Your Best Idea, But It Won't Be Your Last Idea

All of which brings us back to the problems of prototyping. Whenever the technical journals talk about prototyping, they inevitably bemoan the lack of tools that let you design exactly what you want with no effort whatsoever, then click a button and convert the prototype into the code base for the final product. Unfortunately, this sort of thinking reinforces the belief that prototypes are beta versions of the final product, instead of the disposable design ideas that they should be. Part of the reason it's important to prototype rapidly is so you don't grow too attached to an idea you've been working on for a long time. When that happens, we become unable to see our own design's weaknesses, and we never explore other alternatives.

I'll admit, even as I wrote the last paragraph, I winced at the thought of just disposing of some of the prototypes I'm currently working on. After all, they represent my best attempts at solving particular design problems. Throwing them out involves both the mental anguish of having been "wrong" and the effort of creating a new prototype and trying again. If I were designing for myself, I'm pretty sure I would almost always go with my first solution to any problem.

But all of us need to keep in mind that were are almost never designing for ourselves. That's why we need to find representative users, try our designs out on them, then bite our tongues when they inevitably tell us that something is not quite right. In retrospect, the number one problem in user testing is that we don't do it (largely because we're afraid of being found wrong). The number two problem is that we ignore the results (because we can't believe we were wrong). All other concerns about video camera placement and subject selection are largely secondary in the grand scheme of things.

So there you have it: In addition to the growing list of design skills you must have today in order to create a world-class product, we now add learning how to bite your tongue, swallow your pride, and "murder your children." Although the prototype represents your best guess about how to solve a given design problem, you need to leave it out on the ice floe overnight, as it were, exposing it to testing and criticism without being too quick to defend it. If it survives, take it in again and work on it. If it doesn't, start over. Remember, the prototype isn't important—it's the final product that counts.

Till next time,
Doc

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

OpenDoc Your Mind

Why OpenDoc Makes Sense in More Places Than You Think

By Gregg Williams,
Apple Directions *staff*

"Why should I use OpenDoc? My application has nothing to do with documents."

It's a fair question—one that springs from two causes. First, OpenDoc is a new technology, and the implications of a new technology aren't often obvious. (Remember the first time you tried to wrap your mind around how spreadsheets work? Or how to program Macintosh computers?) Second, Apple may have chosen a misleading name when it named its new technology *OpenDoc* (as in "open document architecture"). Yes, the first use for OpenDoc that comes to mind is compound documents, but the OpenDoc architecture is really about *component software*—a much bigger concept that's useful in many more situations.

By using illustrations of hypothetical OpenDoc parts, I hope this article will "open" your mind to thinking in ways that reveal the advantages of OpenDoc. I hope it will cause you to say "Aha! *Now* I see why Apple, IBM, Novell, and Adobe think this is so important."

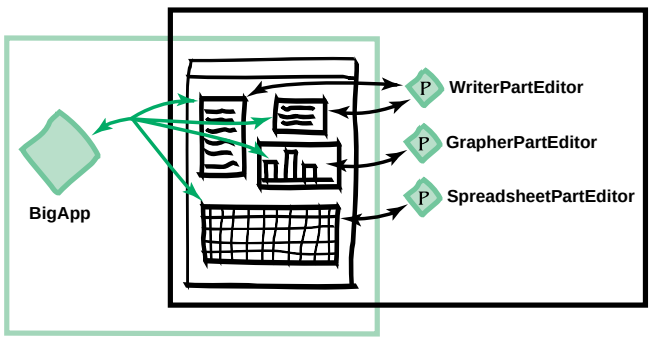
I hope this article will also give you some ideas on how you can use component software (in the form of OpenDoc parts and container applications—applications whose documents can contain OpenDoc parts) in your future development efforts. One caveat: The examples in this article are here to jump-start your own thinking about OpenDoc; please don't read any specific recommendations (about implementation or human-interface issues, for example) into them.

OpenDoc = Component Software

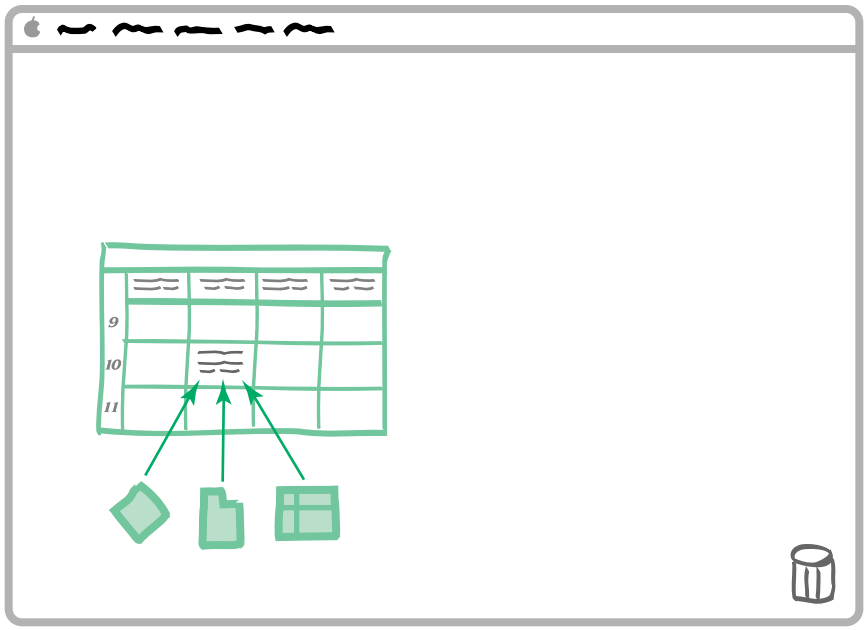
Before I describe a handful of OpenDoc scenarios, let's take a look at why it makes sense to look at OpenDoc as component software. The figure "OpenDoc versus monolithic applications" shows how the two worlds differ.

With a monolithic application, every user event goes to the same (large) block of code. But when the user clicks inside an OpenDoc-based document, OpenDoc figures out which part editor "owns" the clicked area and arranges for the orderly transfer of control to that part editor, which then oversees user interaction with just that area of data.

This is not as huge a paradigm shift from today's graphical-



OpenDoc versus monolithic applications. A monolithic application handles all the user interaction with a document. With OpenDoc-based software, OpenDoc itself routes user events to the particular part editor that should handle it. Part editors are easier to write and maintain because they are smaller and do fewer things.



Dragging and dropping into a scheduler part. Since the scheduler (shown in color) is an OpenDoc part, you can drag items into it, giving the user an easy way to collect everything associated with a certain meeting.

interface computers as you might think—today, the same thing happens every time you click in a window belonging to an inactive application. The only difference from today's paradigm is that, with OpenDoc, the code that (for example) does the graphing and the code that does the word processing are distinctly separated, and the code that referees the switch from one part editor to another—OpenDoc—is system code (which *you* don't have to write). OpenDoc parts are simpler (because they do only one task), and they're activated only when the user event is definitely for them to handle. This makes them easier to write and debug and much less likely to interact with unrelated code—which makes your job easier.

Document? What Document?

OK, imagine we're in a future where all software has been re-engineered as OpenDoc parts and (where it makes sense) every part can have other parts embedded in it. What classes of software would and would not make sense to develop as component software?

Let's start with a challenging example, software that doesn't have any documents at all: networked scheduling software (that is, software that allows multiple people on a network to schedule meetings, see others' schedules, and so on). The scheduler software would be its own part and would appear in its own window, and it might look like the scheduler in "Dragging and dropping into a scheduler part."

Of course (as this figure illustrates), parts can contain other parts, so you can drag any documents, applications, or folders you want into the scheduler part, thereby associating them with a certain meeting or activity. Think of how convenient and useful that would be! Double-click on a meeting (see the figure "Opening a meeting" on page 15), and you can see its details, including all the items associated with it. And, of course, the icons represent the items themselves—you could drag them back out to the desktop, or drag another file into the scrolling list of included files. (Depending on the implementation, the scheduler part could include the actual items or aliases to them.)

But that's not all—you could highlight a meeting, a day, or a group of days, drag the selection into an OpenDoc electronic-mail document, and send an assistant your schedule (see "Sending your schedule" on page 15). Then, since all scheduler parts point to the same networked schedule database, the recipient of that e-mail could invite new attendees, change the time of your meeting, or retrieve the files associated with it!

Client-Server Database

Another type of software that some developers have said is ill-suited to the OpenDoc model is a client-server database. No documents here, either—just a database-query application and a report generator.

Maybe so, but let's look closer. If you implement the query application as a database-query part editor, you can simply tear off a query form from the query-part "stationery pad," fill in the query, send it off, and get back the results—just as you would with a query application. (In the OpenDoc architecture, you use a part editor to create "stationery" for that part. When you double-click the stationery, you get an instance of the part.)

But what if, for example, you want to update your department's employee database? Wouldn't it be great to create a query form, drag it into an e-mail message with an explanatory note, and send it to everyone in your department? When your coworkers get the memo, they could fill in the blanks of the query, click the Send button, and automatically update the employee database. Similarly, you could drag a database report-generator form into a document and generate an employee listing that you can update at the click of a button. The figure "Using database parts in documents" (page 16) illustrates how this might work.

Terminal Emulators

Why make a terminal-emulation application into a part? You're only going to need one of them at a time, right? I can think of one thing that would make a terminal-emulation part more versatile than an equivalent application. The first one is simple, and its advantages are not immediately visible: You could make one document for each telecommunications service you want to connect to. "Big deal," you say. "That's no different from a terminal-emulation application, which also has documents that contain the configuration information for a given service."

But the advantage of a terminal-emulation part is that you can drag it into other parts. Drag it

into your "to do" list, beside the item that says "Get new system enabler from eWorld." Or add such parts to a list of your favorite local BBSs, annotating each entry with information on how to log on to each one. (See "A BBS guide at your fingertips," page 16.) Of course, when you double-click on the terminal-emulation part's small icon, it expands into its own window.

Games

"We've got him now!" you exclaim. "There's no way he can justify writing 'Dungeon Crawl 3-D (in Hack-o-Vision)' as a part!" You're right—there is no real reason to write a classic arcade-style game as an OpenDoc part.

But. . . .

Some games could well be done as OpenDoc parts. Here's how it would work. The part editor would contain the logic for the game—what happens in reaction to user input—so that every game-part "document" (that is, a game in play) would know how to respond to user input. Every visible game element (button, readout, QuickTime movie) would be a part. Since the part editor can dictate the state of every contained part, including its position and visibility, the game's part editor can change the game document's display by making certain embedded parts visible and active and other parts invisible. This approach would work particularly well with CD-ROM multimedia games, which spend much of their time orchestrating the playback of animations and video sequences.

The OpenDoc approach would also work nicely with any kind of game that has "pieces" or "squares." Imagine, for example, a SimCity type of simulation where each piece "knows" how to behave in the "city" (document) it's placed in. This would also open the market for expansion sets ("Oh boy! Shopping malls for SimCity!") and add value by allowing users to customize the game to their preferences (users could, for example, customize the layout of controls in a flight simulator).

Since OpenDoc parts are built using C++, properly constructed parts could be subclassed to create new parts with significantly less extra effort. Also, you could perhaps reuse both the game logic (implemented as the container part) and individual content parts to bring out derivative games in a much quicker time frame than is currently possible.

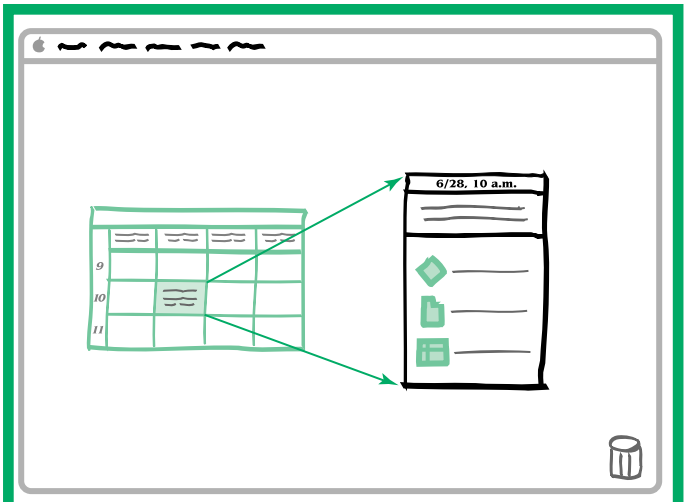
Other Ideas

By now, you should have the basic idea. So here are some additional examples, covered a bit more briefly.

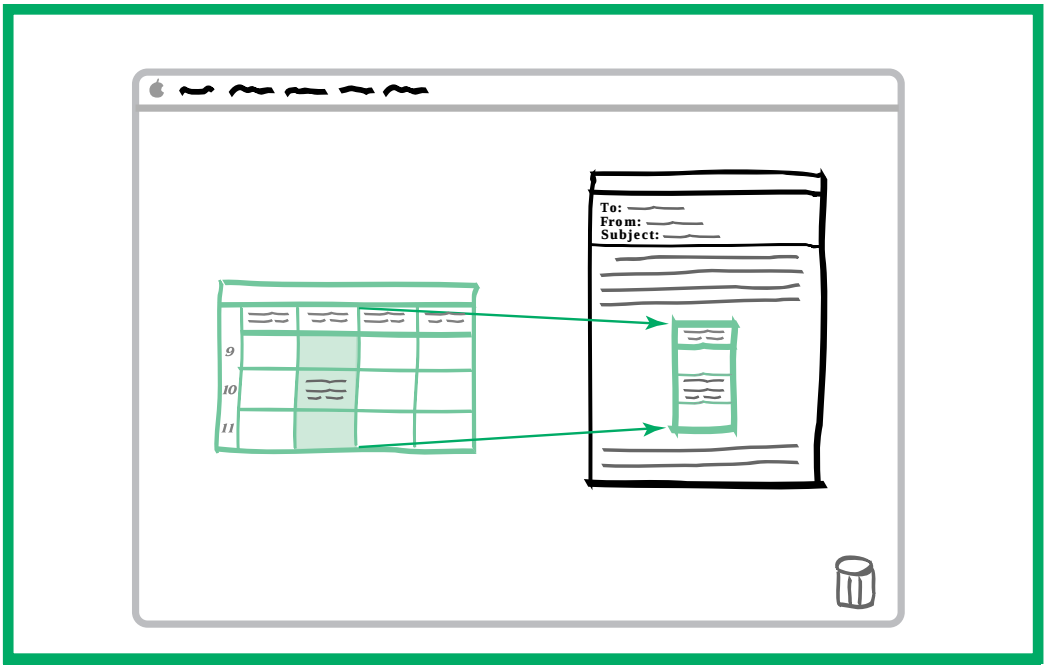
• *Installer.* This is an extension of the file-compressor part idea. Once you make installer software into a part, you can embed an installer into a document explaining how to use the installer. Better yet, embed multiple installers, each configured to install a popular configuration for your product at the click of a button, in a Read Me document. That way, users could read the document and choose the desired installation instead of having to figure out which combination of files to install.

• *PIM.* The case for making a PIM into a part is similar to that of scheduler software. OpenDoc makes it easy to embed any kind of data in any part of the PIM—"to do" lists, address-book entries, project records, client histories—and you can easily share any PIM entry with others.

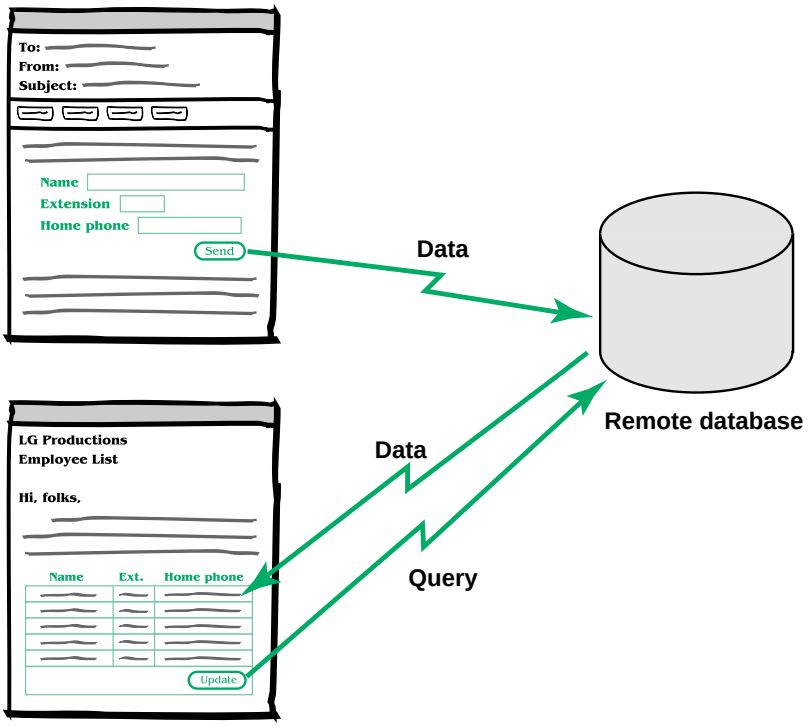
• *Flat-file database.* The major difference between a client-server database part and a flat-file database part is that the latter is self-contained. This means that, with such a part, you can add databases (complete with their report generators, data-entry forms, and searching capabilities) into all your documents. Imagine adding a FileMaker Pro–like database into one slide of a presentation program and being able to call up



Opening a meeting. Double-clicking on a meeting reveals the items associated with it.



Sending your schedule. Since the scheduler is a part, you can embed it in other parts—including, as shown here, a piece of OpenDoc-enabled e-mail.



Using database parts in documents. In this example, the database-query part (in color) is sent to employees through e-mail; when they fill it out, the part updates the remote database. Similarly, the report-generator part (also in color) queries the database to get its data.

different data while showing that slide.

• *Software development environment.* A part-based software development environment could be very much like today's project-based environments, but much more versatile. A software project would be a container part. It would have its own window, and you would drag source files into it to add them to the project. But it could also treat any nontext part as comments, allowing you to annotate your project with pictures, sound, or even executable test cases to be run during debugging. You (and others) can extend the environment by adding new parts—like resource editors and interface builders.

• *Desk accessories.* Why would you want to make these into parts? (You can probably guess the answer by now.) Take a calculator part that stores its calculations on a built-in "tape" for review—add that part to a document, and whoever reads it can review your calculations. With a "find-file" part, you could make multiple "parts," each of which stores an often-used query. You could also add them to documents (with explanatory text) and send them to others. (For example: "Mary, John, and Emilio—I'm looking for presentations on last year's budget. Please click the button below and send me whatever you find.")

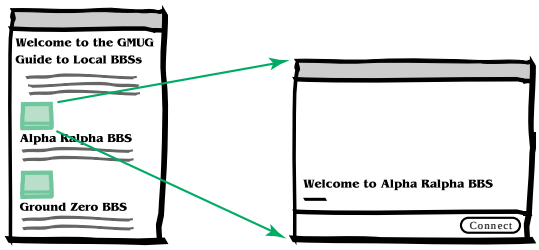
Document as Container

When I go to the gym, I don't pick up my shoes in my right hand, my shorts in my left, my water bottle under my right arm, my T-shirt around my neck . . . you get the picture. Instead, I unzip my gym bag, load in all those items (plus a towel, a book, and several other things), and can carry everything with me—still keeping one hand free! And, just as I want a gym bag for my exercise equipment, I want a backpack to put stuff in when I go out, a folder when I want to organize my paperwork, a sack when I buy things at a store—there's a metaphor for "container" in every human context.

Unfortunately, our options for "container" in the world of electronic information are quirky and limited—on a floppy or removable cartridge disk, in certain applications that allow files to be "attached" to a document, and in archive files that store (and compress) multiple files into one. These solutions are limited, and they do nothing to relate the various pieces of content to each other.

OpenDoc promises to change computing by making the document into an infinitely customizable container for information. And it isn't just a container that holds items together (as does my gym bag)—it allows you to interrelate its contents meaningfully by how you arrange and format them.

So maybe *OpenDoc* isn't such a bad name after all: It makes documents open to any kind of data—or active behavior (in the form of programs that do something when you click on them)—you want to include. The opportunity for you is selling your part editor (or container application) to every person who wants to tap into this extremely flexible and useful way of dealing with electronic information. ♣



A BBS guide at your fingertips. The colored icons are terminal-emulator parts. Double-click on one, click Connect, and the part automatically connects you to an electronic bulletin board system (BBS).

Developer University Schedule

To help you find the Macintosh and Newton courses you need, each month *Apple Directions* publishes the latest Developer University (DU) schedule. DU is Apple Computer's technical institute for developers, offering training in how to program with key Apple technologies. Most courses are offered at Apple's R&D Campus in Cupertino, California, although several take place at the Portsmouth, New Hampshire, training facility.

To receive more information, including a catalog and detailed schedule, or to register for a class, contact the Developer University Registrar by phone at 408-974-4897 (select 2 when you reach the phone tree), by fax at 408-974-0544, or by AppleLink at DEVUNIV. A list of short descriptions and a schedule is also available on AppleLink (path—Developer Support:Developer Services:Apple Information Resources:Developer Training).

Advanced C++
(three days—$1,000)
Feb 27–Mar 1, Cupertino

Apple Events/AppleScript Programming
(five days—$1,500)
Feb 6–10, Cupertino

Creating Apple Guide Help Systems
(four days—$1200)
Feb 7–10, Cupertino
Mar 14–17, Cupertino

Designing Object-Oriented Frameworks
(three days—$900)
Feb 7–9, Cupertino
Mar 14–16, Cupertino

Intermediate Macintosh Application Programming (IMAP)
(five days—$1,500)
Feb 13–17, Cupertino
Mar 20–24, Cupertino

Introduction to OpenDoc
(one day—$300)
Dec 16, Cupertino
Feb 17, Cupertino
Mar 31, Cupertino

Macintosh Debugging Strategies and Techniques
(three days—$900)
Jan 30–Feb 1, Portsmouth
Mar 6–8, Cupertino

Macintosh Programming Fundamentals (MPF)
(five days—$1,500)
Dec 5–9, Cupertino
Jan 30–Feb 3, Cupertino
Mar 6–10, Cupertino

Newton Programming: Essentials
(five days—$1,500)
Jan 30–Feb 3, Cupertino
Mar 13–17, Cupertino

Newton Programming: Extended Topics
(five days—$1,800)
Feb 6–10, Cupertino
Mar 20–24, Cupertino

Newton Programming: Communications
(three days—$1,175)
Feb 13–15, Cupertino
Mar 27–29, Cupertino

Newton Technology Overview
(one day—$325)
Dec 16, Cupertino
Feb 17, Cupertino
Mar 31, Cupertino

Object-Oriented Fundamentals
(five days—$1,500)
Feb 27–Mar 3, Cupertino

PowerPC Boot Camp
(four days—$1,600)
Nov 14–17, Cupertino
Dec 12–15, Cupertino
Feb 14–17, Cupertino
Mar 21–24 , Cupertino

Programming With MacApp
(five days—$1,500)
Dec 5–9 , Cupertino
Jan 30–Feb 3, Cupertino
Mar 13–17, Cupertino

Programming With OpenDoc
(four days —$1,600)
Dec 5–8, Cupertino
Feb 27–Mar 2, Portsmouth

Programming With QuickDraw GX
(five days—$1,600)
Feb 13–17, Cupertino

Scripting With AppleScript
(two days—$600)
Dec 15–16, Portsmouth
Mar 23–24, Cupertino

continued from page 10

supported by Sound Manager 3.0 for compression, decompression, custom hardware support, and more.

This issue of *develop* also tells how to improve QuickDraw GX printer driver performance and how to support text tracks in QuickTime movies. You can read about PowerPC branch prediction, garner useful tips of all kinds from our Q&A sections (for Macintosh and Newton), and test your debugging skills against a couple of real dogs.

If you want answers to your development questions, and a good time while you're at it, *develop* is the place to look. Issue 20 and the code it describes are on this month's Developer CD (along with all back issues of *develop*), and the printed copy is available by subscription through APDA. Enlightenment was never easier.

Caroline Rose
Editor, develop

CD Highlights

continued from page 11

Manager to generate compressed picture files. It uses QuickTime to compress images according to the specification provided by the user through the standard compression component.

- Thumbnail Test: This snippet shows how to use CopyBits to copy an image to a black-and-white "thumbnail" picture.

Coming Next Month

Next month's CD will include localized versions of System 7.5 and other system software, a sample application using the Subspace Manager as described in *develop* Issue 7, and more.

Alex Dosher, Developer CD Leader

Business & Marketing

Market Research Monthly

What Educational Customers Want: A Profile of the Market

If you're interested in exploiting the Macintosh computer's clear lead in the U.S. K–12 educational market (see the Editor's Note on page 2), you need to know what customers in that market want out of their software. This month we provide analysis and data from Apple Computer, Inc., and other sources to help you meet the specific and general needs of K–12 customers.

Specifically, what schools appear to want most can be said in a word: multimedia. Multimedia technology is proliferating at an unprecedented rate in school settings, and school users increasingly demand software that ships on CD-ROM and employs a variety of media. Virtually the entire Macintosh educational product line ships with a built-in CD-ROM drive; only the lowest end (under $1,000) models ship without one. Currently, four out of every five Macintosh computers purchased by U.S. K–12 schools include CD drives, according to Apple internal data.

Also, there's widespread demand for multimedia titles in more subject areas than ever before, according to Quality Education Data, Inc. (QED). In a study of multimedia-using school districts in the United States in 1993 and 1994, QED found that science multimedia titles were used in the most districts—68 percent of them—followed closely by social studies (65 percent).

Specific Areas of Opportunity

The QED study indicates that there may be specific opportunities to develop multimedia titles for some subject areas—namely art, math, music, industrial arts, and foreign language. In each of those subjects, significantly higher percentages of schools want multimedia software than the percentage of schools actually using it at present. For example, only 35 percent of the schools covered by the study use art multimedia software today, while 51 percent would like to use it in the future.

This gap between current use and future desire suggests that there may be pent-up demand for multimedia software in these subject areas. For complete results of the study, see "Multimedia-Using School Districts, on page 19. One warning: Before you leap into developing products in these areas, we strongly suggest you conduct your own, more detailed customer research.

This data is reprinted with permission from the report *QED Educational Technology Trends, 1993–94.*

General Trends in the Classroom

Moving from the specific to the general, multimedia is not the only category in which schools are looking for new software. Two groups at Apple, the Apple Classroom of Tomorrow (ACOT) and Evangelism, are devoted to finding out how educators, parents, and students want to use computers in educational settings. Most generally, they've found that schools are looking for software tools that can help students construct their own learning processes. Traditionally, educational software has been used for drill and practice purposes—for example, helping students learn to read and perform basic mathematics. Increasingly, educators and students are looking instead for creative tools and problem-solving aids.

ACOT and Apple Evangelism have found that schools are most interested in software that falls into the following general categories, in addition to multimedia:

- communications: software that helps students access information over the Internet is in particular demand
- collaboration: students often work together on projects, and software that helps them collaborate is highly valued
- presentations: especially in higher grades, students use personal computers to present and defend their ideas
- simulations: software in this category helps students create "real-world" situations and learn from them

What Makes Good Software?

Apple Evangelism has also talked to teachers, parents, administrators, and children to determine the characteristics that make for quality software, regardless of category. Here are some of the conclusions they've drawn about what customers in the U.S. K–12 market are looking for:

- Successful applications need to be integral to class curriculum and help teachers and students further their specific educational goals.
- K–12 customers prefer software that engages many senses at once (sight, hearing, and so on).
- The most successful K–12 software employs a "natural" interface that requires little or no learning by the user.
- It's becoming increasingly important for K–12 software to be released in multiple languages, because of the growing number of students who speak languages other than English at home.
- Many districts consider it a plus if your product can be used by children with disabilities.

Inside This Section

• CD-ROM is fast becoming the delivery medium of choice.

• Software for schools should also be usable in the home, so children can continue their education outside the classroom. Software that's useful both in home and school learning situations will also generate sales in the home learning market. (See "Home Learning: The Fastest Growing U.S. Software Market" on page 19 of the July 1994 *Apple Directions* for more on this market.)

You'll want to build these characteristics into your K–12 products to give them maximum appeal to customers.

Configuration Data

Finally, to attain the greatest sales in any market, you need to write code that supports the configurations used by the majority of customers. Apple installed base research indicates that the vast majority of Macintosh computers in K–12 settings run System 7, although System 7.5 is quickly being adopted. The following is Apple's configuration data for the U.S. K–12 market:

• Microprocessor: 85 percent of the K–12 installed base uses at least a 68030 processor, although 68040 processors are appearing at increasing rates.

• Color: 76 percent of the installed base uses 8-bit color or greater.

• Monitor size: 76 percent use at least a 14" monitor.

• CD-ROM drive: 50 percent of the installed base uses a CD drive, while 80 percent of Macintosh systems currently sold to the K–12 market include a CD drive.

• Memory: 77 percent use 4 MB; 30 percent use 8 MB.

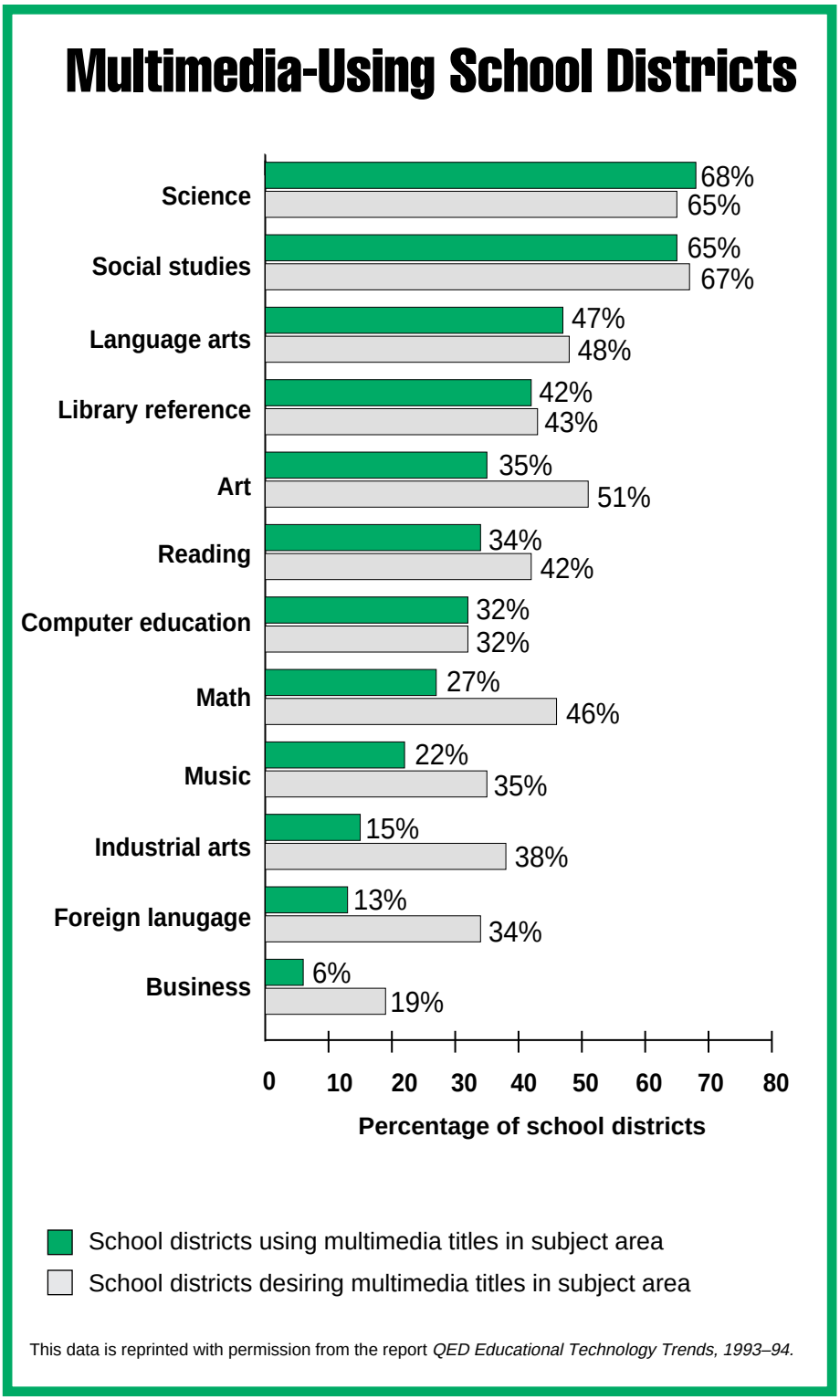
To conclude, the target platform you'll want to design your K–12 products to work with employs at least a 68030 processor, 8-bit color, a 14" monitor, a CD-ROM drive, and 4 MB of memory.

Power Macintosh computers have yet to enter the K–12 market in a significant way, although Apple expects that to change rapidly, partly because of the release of the first Power Macintosh Performa systems (see the news story on page 8). You should start planning now to port your educational applications to run in "native" PowerPC mode before schools begin to adopt PowerPC technology in big numbers.

Kids Say the Darndest Things

This is a brief sketch of what customers in the U.S. K–12 education market are looking for. While we hope this information can point you in the right direction, we can't urge enough the need for you to collect you own data, whether by going out and talking to potential customers yourself or by hiring market research firms to do it for you. In the words of Art Linkletter, "Kids say the darndest things": we can guarantee you that they—and their teachers and parents—will tell you a great many specifics about how they'd use computers to help the learning process.

For more data on the educational market, you can contact QED at 1600 Broadway, 12th Floor, Denver, CO 80202; phone: 800-525-5811; fax: 303-860-0238; AppleLink: QUALITY.EDUC.♣



This data is reprinted with permission from the report *QED Educational Technology Trends, 1993–94.*

Marketing Feature

OpenDoc and Your Business

Six Developers Talk About How OpenDoc Will Give Them a Competitive Edge

By Kris Newby

No one can definitively say what the advent of the OpenDoc component software architecture will do for your business, but in this article we provide you with the next best thing—the predictions of developers who are well on their way to creating the first OpenDoc-based applications.

The question on the top of most developers' minds is "Can I afford the time and effort needed to change my applications to OpenDoc?" We interviewed six developers who are currently evaluating OpenDoc, and their answer was "We can't afford not to." In the increasingly crowded software market, these developers feel that OpenDoc will help them more effectively compete by

- making it easier and less expensive to develop software for multiple hardware platforms
- enabling them to create better products in less time
- expanding market opportunities

Most of the developers that we talked to are creating OpenDoc components called "part editors." Part editors, the content building blocks of OpenDoc documents, are usually compact and focused on specific user needs. Users can drag and drop these parts into any OpenDoc "container" application when they need to add specific content or functions to a document.

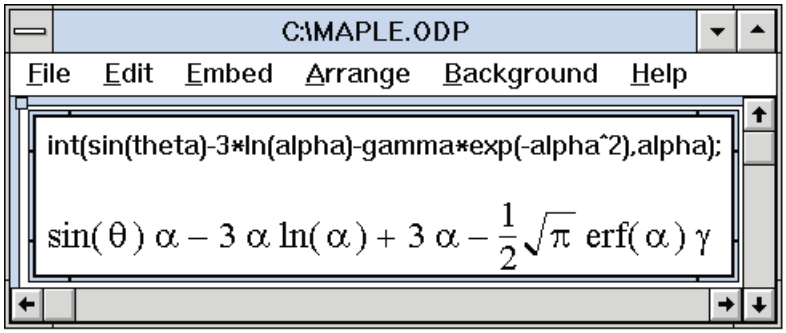
Another developer, Adobe Systems, will be creating "part viewers" for some of their major applications like Photoshop and Illustrator. Part viewers allow anyone to view and print OpenDoc content, even if users don't have the complete part editor or application installed on their systems.

Read on for more details about these developers' upcoming OpenDoc-based products, as well as analyst predictions on how OpenDoc will affect software pricing (page 21) and a few reasons why OpenDoc is better for your business than the competing architecture, Microsoft OLE (page 23).

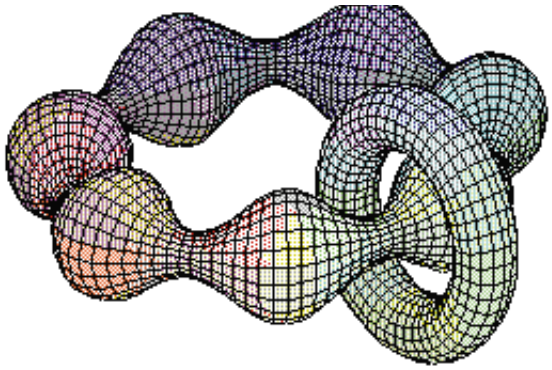
WMS: Multiplying Customers With Mathematics Parts

Waterloo Maple Software (WMS) is perfectly positioned to benefit from developing and marketing OpenDoc parts. First, it is focused on doing one thing very well—its Computer Algebra Systems (CAS) and scientific visualization engines are in the hands of more than 500,000 users and OEMs (original equipment manufacturers) worldwide. And second, WMS leverages its expertise by porting its products to 30 different computing platforms, including Macintosh, Windows, and a variety of UNIX versions.

WMS's flagship product, Maple V, is primarily used by educators, mathematicians, physicists, chemists, and engineers. WMS also develops and sells Theorist, a compact, intuitive Computer Algebra System, used in industry, colleges and high schools. Its equation editor package, Expressionist, is



A Tube Plot: Maple Graphics OpenDoc Part



It took MWS developers about a week to create their first OpenDoc Windows-based mathematics part, shown here. This company expects that OpenDoc will significantly grow its market and reduce cross-platform development costs.

linkable to many popular word processors through EGO (WMS's own Embedded Graphic Object programming interface) and OLE technologies. And WMS recently launched MathEdge, an application development library that enables linking to the math engine heart of Maple V.

This Canadian firm anticipates that OpenDoc will help increase its market share by making it easier for users to add their mathematics parts to virtually any document, on any platform.

And secondarily, WMS appreciates the fact that OpenDoc will reduce the time and cost required to develop, debug, and maintain its numerous product versions.

"We created our first functional OpenDoc mathematics part in about a week, and right now we're enhancing it with graphics support. Soon we'll convert our Expressionist product into an OpenDoc part," said Subhashis Mohanty, a WMS software developer. "The ease with which OpenDoc interoperates with other applications will expand our market and allow us to reach a wider class of users who need a robust math engine."

In summary, WMS expects that OpenDoc will help the company's business by enabling the following benefits:

- *Market share growth.* As more applications become OpenDoc-savvy, more users will be able to integrate WMS's parts into a larger number of popular word-processing and spreadsheet applications, thereby expanding its market share.
- *Reduced cross-platform development costs.* WMS anticipates that OpenDoc's architecture will enable the company to move its mathematics parts to multiple platforms faster and less expensively.

Lumina: Staying Competitive With Drag-and-Drop Technologies

"With OpenDoc, new technologies are just a drag-and-drop away," said Brian Arnold, director of software development at Lumina Decisions Systems, Inc. "Last year I panicked about supporting AOCE. This is a simple API (application programming interface), but as a small developer, I just didn't have the time to implement it. I also put off adding sophisticated drawing tools and arbitrary embedding of QuickTime movies because of lack of time. Now with OpenDoc, I can just drop these new functions into my part editor—I get them for free. And I won't have to worry about compatibility with older or newer product versions."

Lumina's visual decision-modeling software, called *DEMOS,* enables engineers and consultants working on Macintosh, Windows, and UNIX platforms to quickly assess the risks associated with complex business decisions. NASA, for example, uses DEMOS to decide how to schedule turnaround tasks between space shuttle missions, weighing the estimated completion times to determine the optimal allocation of resources.

Arnold also thinks the OpenDoc architecture will help him develop applications faster: "My application consists of multiple-linked file pseudo-databases," he said. "Windows and views are hierarchical, many in number, and stored via database-like APIs. After spending a week with OpenDoc, I realized that OpenDoc's architecture is much better suited for my type of product than MacApp's. Using OpenDoc, creating linked hierarchies of my application's components will be more transparent, and creating multiple views will be less of a kludge."

Overall, Lumina looks forward to these two OpenDoc benefits:

- *Easier adoption of new technologies.* Lumina will be able to drop new features into its products rather than having to create the "glue" that links these features. Thus, the company will be able to ship product enhancements in less time, with fewer programming resources.
- *Improved customer satisfaction.* Lumina will be able to focus on delivering the best possible decision-modeling software to its customers, rather than having to spend time creating and maintaining features peripheral to the company's expertise. In addition, customers can choose to use Lumina's graphing facility or replace it with their favorite third-party graphing package.

Adobe: Building a Following With Part Viewers

In September, Adobe announced that it would become a full

Software Pricing in an OpenDoc World

One of the questions developers have about OpenDoc and component software is how it will change the software pricing.

Pieter Hartsook, the respected computer industry analyst, comments on this question: "Initially OpenDoc shouldn't have much effect on the way software is priced and distributed. Software components primarily will be used internally within monolithic shrink-wrapped applications, and users will continue to purchase products as they do today. Perhaps three years from now users will start seeing a reduction in software prices, as this technology helps developers more efficiently create and test parts."

Jerry Michalski, managing editor of *Release 1.0,* believes that OpenDoc will create opportunities for smaller developers who don't want to develop monolithic applications. Michalski said, "Half of the software that I see coming through our office shouldn't be full-fledged shrink-wrapped applications—they should be features that can be added to a larger multifunctional application. Many of these special functions—like groupware tools, brainstorming aids, bibliography facilities, and indexing tools—would be more successful as something people could drop into a major word processor."

How does Michalski think OpenDoc will affect software prices? "OpenDoc component prices will be lower than those of monolithic applications, but the net effect of component technology should be to increase revenue opportunities for developers. The components will be cheaper to buy, and people will purchase more of them " said Michalski. "And my guess is that some of the monolithic applications developers will welcome the opportunity to break their larger applications into components. This will enable them to make money off some of the clever functions buried in their 'do-all' applications."

The difficulty of getting products on retail shelves is a problem faced by most developers these days, and an ancillary benefit of OpenDoc is that it opens a less expensive avenue of distribution for compact OpenDoc parts—electronic delivery.

"In the future, it will become more commonplace for users to call in a software order and have an update automatically installed on their computers via the Internet," said Michalski. "And though the try-and-buy product CD concept has gotten off to a slow start, I see an increase in the use of this delivery in the future."

Component Integration Laboratories (CI Labs) sponsor, signaling its strong support for the OpenDoc standard.

"We think OpenDoc is a powerful way to achieve cross-platform component software," said John Warnock, chairman and chief executive officer of Adobe Systems, Inc. "Adobe will be developing companion OpenDoc parts for our major video and graphics applications. These parts will allow basic display and printing of documents in the content formats of Adobe™ Illustrator, Adobe Photoshop, and Adobe Premiere software."

Jerry Barber, director of technology integration at Adobe, elaborated further on their development plans at a recent OpenDoc briefing: "We like the fact that OpenDoc is open and that it interoperates with OLE. There is clearly synergism between OpenDoc and our Acrobat document-viewing software, and there is the potential to create OpenDoc-based authoring tools."

Adobe anticipates that OpenDoc will help the business's bottom line by providing these advantages:

• *Easier multiplatform development.* Adobe is a large software development company that has to quickly release products on multiple platforms. OpenDoc will make this engineering task significantly easier and faster.

• *Increased sales.* OpenDoc offers Adobe the potential to increase sales to new customers. Its part viewers (referred to as "companion parts" by Adobe) will make it easier for users to incorporate their content into any type of document. Part viewers, in essence, become product demos that may persuade new users to purchase their full-featured applications.

Aladdin: Keeping OpenDoc Documents Bottled Up

"In the short term, the OpenDoc architecture will provide us with more opportunities to integrate our technologies with third-party products," said Leonard Rosenthal, director of advanced technology at Aladdin Systems, Inc.

Aladdin specializes in data compression and communications software that helps users save disk space and speed up the transfer of data across networks. (Many Macintosh users are familiar with their StuffIt Deluxe compression utility.) Rosenthal and his team believe that OpenDoc will increase the need for compression utilities, because as users begin integrating more parts into OpenDoc documents, files will get increasingly large and unwieldy.

"Our first OpenDoc-specific product will automatically compress parts within OpenDoc documents, helping users delay the purchase of larger hard drives or more memory," said Rosenthal. "We're also developing a new class of product that we call an *enclosure part.* Enclosure parts will enable users to embed compressed files within an OpenDoc document, providing readers with a convenient method of accessing reference materials. For example, a user could create an OpenDoc-based product data sheet and embed a detailed compressed product benchmark report within it."

So over time, OpenDoc will help Aladdin expand its business because of the following:

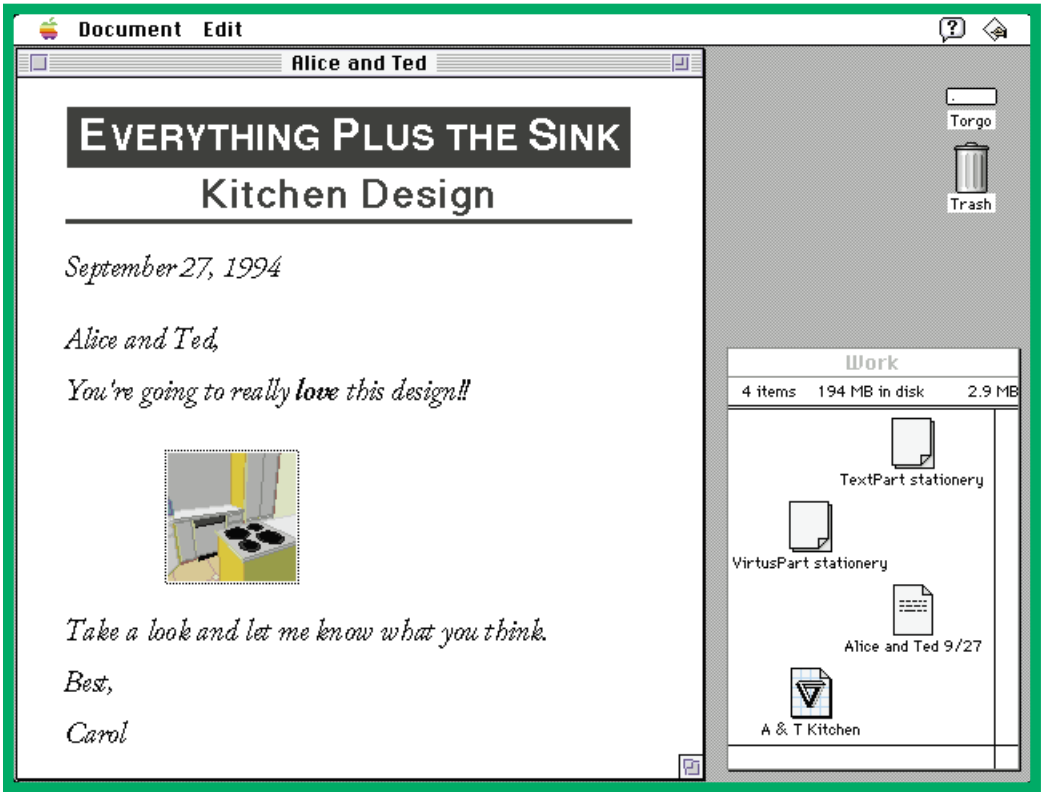
• *New market opportunities.* OpenDoc will create two new market needs that Aladdin is qualified to meet: the need for a compression utility to reduce the memory footprint of large OpenDoc documents and the need for an "enclosure part" that provides a convenient way to efficiently store reference information within a OpenDoc document.

• *Expanded market.* Because of the compatibility between OpenDoc parts, Aladdin will be able to license their OpenDoc compression utility to other developers who want to bundle this function within their part or container application. And easier cross-platform development will enable the company to sell their product to a wider audience of computer users.

Route 66: Plotting a Parts-Based Strategy

Job van Dijk, president of Route 66, Geographic Information Systems B.V., sums up the most important aspect of OpenDoc to his business's profitability: "OpenDoc enables us to more quickly and consistently release new versions across Macintosh and Windows platforms."

Route 66, based in the Netherlands, currently sells a wide variety



With this prototype 3-D visualization part from Virtus, a designer can drop a rendered drawing of a new kitchen design into a text-based proposal, enabling the clients to take virtual walk through the kitchen on their computer.

of products that serve the consumer Geographic Information Systems (GIS) market. Its cornerstone product, ROUTE 66, is a route planner that finds the shortest, fastest, or least expensive route between chosen locations in the United States, Europe, or Japan. The modular, object-oriented nature of OpenDoc will work well with its installed base marketing strategy, because it allows users to easily drop new map modules into the ROUTE 66 product and use this product within container applications.

"We already have working OpenDoc applications in our labs, and we plan to release an OpenDoc-savvy suite of applications upon introduction of this technology," said van Dijk. "Using OpenDoc, our customers will be able to zoom in and out of maps and calculate routes while working in faxable documents. For example, a manufacturer could drop a ROUTE 66 part into their fax-on-demand product information system that provides a prospect with a map to their nearest retail outlet. Or a travel agency could fax a holiday traveler a completely customized route description from their home to that of a hotel. It opens up a whole new range of applications for end-users."

OpenDoc provides Route 66, a small international developer, with advantages that help it compete with larger developers. The two main advantages that Route 66 expects to benefit from are these:

• *Reduced multiplatform development costs.* Route 66 estimates that OpenDoc will cut the company's multiplatform development costs by 25 percent and will significantly reduce the time it takes to bring these versions to market. Reduced development costs will help Route 66 price its product low enough for a consumer audience and for markets outside the Netherlands.

• *Expanded market.* Route 66 anticipates that OpenDoc will enable its product to be used more widely by users, because it will make it so much easier to use their maps within popular word processors and communications applications. For example, a conference planner could use the part to quickly add a convention center map to an attendee mailing.

Virtus: Turning Virtual 3-D Parts Into Reality

"OpenDoc will make it easier for small developers like Virtus to get new technologies into the hands of more users," said David Easter, Virtus director of technologies. "We hope that this new standard will enable our upcoming three-dimensional (3-D) visualization part to be used widely to create innovative types of 3-D environments such as OpenDoc parts containers and interactive museums."

Virtus specializes in 3-D modeling software that lets users "walk around" objects and spaces on their desktop computers. The company currently has a prototype OpenDoc part embedded within a fictitious kitchen designer's bid document that allows a client to walk through a virtual remodeled kitchen. This part lets a client try out different design variables, such as kitchen countertops and fixtures. Virtus anticipates that this part will initially be used primarily by architects, builders, and interior designers.

"OpenDoc will allow us to concentrate on what we do best—3-D visualization software—then combine these technologies with other developers' products to create complete solutions," said Easter. "Larger companies unwilling to take the risk of developing new technologies will be able to comarket products with smaller companies like Virtus. They benefit from our innovation, and we benefit from their market leverage."

Besides allowing Virtus to focus on its area of expertise, OpenDoc will help the business in these areas:

• *New market opportunities.* The inherent compatibility between OpenDoc parts will make it easier for Virtus 3-D technologies to be "dropped into" established products, speeding the proliferation of Virtus products in the market. The company also anticipates that its technologies will catalyze the development of entirely new categories of products, such as interactive 3-D museums.

• *Comarketing opportunities.* Interoperability will also make it

Why OpenDoc Is Better for Your Business

Today there are two component software architectures vying for developer support—CI Labs' OpenDoc and Microsoft's OLE. OLE 2.0, like OpenDoc, is a software architecture that enables applications created by different developers to work together. And though on the surface these architectures appear to achieve the same end, there are some fundamental differences that make OpenDoc a better choice for your business.

Here are some specific business advantages that OpenDoc provides over OLE 2.0:

• *OpenDoc is an open, cross-platform technology.* OpenDoc is a true cross-platform architecture that enables you to leverage your development efforts across Macintosh, Windows, OS/2, and UNIX platforms. In contrast, OLE's scope is narrower—it was designed to allow applications to interoperate with other compliant applications written for Microsoft Windows. And besides, who would you rather have managing an important development standard—a competitor or an industry-wide coalition?

• *OpenDoc includes OLE 2.0 compatibility.* By supporting OpenDoc, you'll be better positioned to sell products to users working in real-world multiplatform environments, including Windows users. OpenDoc's ease of cross-platform development lets you hedge your bet on which hardware installed base will come out ahead in the next decade. (For more details on OpenDoc/OLE compatibility, see the last half of the cover page article, "OpenDoc *Is* Cross-Platform," *Apple Directions,* November 1994.)

• *OpenDoc facilitates a faster product development cycle.* Because OpenDoc is a high-level architecture, it allows your developers to work more quickly and efficiently. OLE 2.0, on the other hand, has complex interfaces, forcing developers to deal with low-level details.

• *OpenDoc can provide a better user experience.* OpenDoc offers users with some important features that OLE is missing—such as multiple active objects and nonrectangular part shapes.

Of course, OLE supporters are quick to point out that OLE is shipping, and OpenDoc is still in the beta phase. Though true, OpenDoc still appears to be the best *long-term* proposition for your business. And since OpenDoc includes OLE compatibility, it's never too late to switch.

Perhaps Pieter Hartsook sums it up best: "OpenDoc isn't just hand waving in response to OLE. OLE may be here now, but OpenDoc is a serious development effort, and it's on the fast track from virtual to reality."

easier for smaller developers with complementary products to work together to offer superior part "bundles." Virtus sees opportunities in combining forces with established 2-D computer-aided design developers.

Back to the Future

In the near term, the OpenDoc standard offers your business a way to reduce development costs and reach more customers. Looking ahead, OpenDoc provides future programmers with a fundamentally better way of creating software that minimizes redundant efforts and rewards innovation.

But like all progress, it takes some work on your part. And a "standard" only works if everyone plays. No one can say exactly what changes OpenDoc will bring about, but Apple considers it an important foundation to the future of software development. Our message to those considering OpenDoc adoption is best summed up in the words of Antoine de Saint-Exupéry, the heroic French aviator and author of *The Little Prince*: "As for the Future, your task is not to foresee, but to enable it." ♣

Kris Newby is a technical communications consultant and freelance writer based in Palo Alto, California.