The Developer Business Report *November 1994*

AppleDirections

Inside This Issue

Apple News

Mac OS Licensing Update

We can't say for sure when it will happen or whose will be the first, but one fact is certain: Your software will soon run on what Apple hopes will be an army of Macintosh "clones" created by companies that license the Macintosh operating system.

Apple Computer, Inc. recently outlined how it plans to license the Macintosh operating system to other personal computer vendors. Apple is delaying specific announcements to be sure that the licensing program is being carried out as effectively as possible. Also, specific licensing announcements will be made not by Apple but by the licensees themselves, in whatever time frame they deem appropriate.

The company will take what it calls an "expanded markets" approach, licensing initially to companies with strengths that are complementary to Apple's, and then, over time, licensing more broadly to a wider range of vendors. These licensing activities are planned to result in a variety of personal computer vendors bringing to market Macintosh operating system–based personal computers that run Macintosh applications. The first Macintosh "clones" to result from other companies licensing the Macintosh operating system could reach the market by the second half of

please turn to page 9

Strategy Mosaic

OpenDoc *Is* Cross-Platform

By Gregg Williams, Apple Directions *staff*

It's no secret: You're being asked to do more—a lot more—and you just don't see how you can get it all done. You need to support new technologies to stay competitive, but doing so slows development. At the same time, you're told you need to get your products to market faster—which you can't do if you support new technologies or add new features to your products. You could sell more if you could tailor your products to different market segments, but that means more versions of your product—which also means more development time and more expense. On top of all that, you could make more money if your products were available on multiple platforms, but that means hiring more programmers and developing expertise on multiple platforms, not to mention more expense, more delays, an organizational nightmare, destruction, doom, and the end of civilization as we know it. *It's just not possible.*

Or is it?

Consider the following counter example. Someone at Apple wrote a simple drawing program as an OpenDoc part in 7,993 lines of code. When he converted it to Windows (where it would work with any Windows program that accepts OLE objects), he found that only 22 lines of code were platform-specific—that's 99.7 percent code reuse. As you can see

please turn to page 4

AppleDirections

Volume 2, Number 11

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
Dave Bice, Peter Bickford, Alex Dosher, Gary Little, Kris Newby, 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

Pizzas Over the Internet

During a recent extended vacation from Apple (where we call it a *sabbatical*), I expected to hear people talking about PowerPC, Pentium, Chicago, and so on. Instead, when the conversation turned to computing, the big questions were not "How fast is your computer?" and "Have you made the jump to RISC?"; instead, people were asking each other, "Are you on the Internet?"

Aside from reinforcing something I've believed for quite a while, namely that most people care more about what they can do with a computer than about the computer itself, these conversations made me realize just how much of a phenomenon the Internet is. It also made me think that if I were a developer, I'd be thinking of a way of capitalizing on that (abhorrent as it may be for some of you to think commercially about one of the bastions of hacking).

From what I've seen recently, the Internet is the newest icon of popular culture. Everybody wants to be part of it, and I don't just mean the personal computer elite. Here are a few examples:

- A mainstream (that is, noncomputer) bookstore I visited dedicated an entire front table to books about the Internet.
- I got together with a group of old friends, among whom the Internet is quickly becoming the communications medium of choice. Handwritten letters just take too long to compose and send, we're tired of trading voice mail messages, and we see each other far too infrequently to wait until the next time we can talk in person. Instead, we send short messages over the Internet.
- Then there's the article in the September 23 edition of the *Wall Street Journal* about Mosaic, the graphical-interface Internet browser originally used by a few scientists to search the Internet and now employed by more than 1 million people. The *Journal* called Mosaic "the electronic sales booth" and talked about how it's being used for tasks like receiving Rolling Stones concert tour information and ordering takeout pizza. Vinton Cerf, one of the Internet's founding fathers, called Mosaic "the single most important factor in commercialization of the Internet," and one report cited in the article predicted that Internet commerce could exceed $100 million next year.

A friend who works as an agent for computer book authors recently said to me that she wished she'd been working on a book about the Internet a year ago to capitalize on the current craze. I told her that I didn't think the craze has yet reached anything near its peak.

People on the Internet today are, to a great extent, people like you and me who have some past experience with computers (if we're not out-and-out technophiles). But many of the people who want to get on the Internet are like my 70-year-old aunt who has never touched a computer but is talking about buying one just to see what all the e-mail fuss is about. There are many more people like my aunt who have never used the Internt than the current technophile crowd, and they don't want to do technical research—they want to shop, gossip, be entertained, and get their laundry done.

My point in telling you this is to suggest that there is a vast opportunity for creating and selling Internet products, not for experienced users but for "the rest of us." Agents to help people search and organize the Internet's stores of information, front ends to hide its basic complexity, solutions to help technophobes configure their systems and log on, new compression products to enable fast transmission of large audio and video files, instructional materials that make Internet communications easy for the neophyte—these types of products will all be appealing to new computer and Internet users.

Who knows? Maybe Internet communications (of all things) will finally make the personal computer that "must-have" consumer item. Judging from everything I've been seeing lately, that idea may not be so far-fetched.

Paul Dreyfus
Editor

P.S. My sincere thanks to Technical Editor Gregg Williams, and the rest of the staff, who produced two marvelous issues of *Apple Directions* in my absence.

IndustryWatch: News & Perspective

Decisions, Decisions

Prepared by the Apple Directions *staff*

Dancing to the OpenDoc Beat

Adobe and Lotus have joined Component Integration Laboratories, Inc. (CI Labs), the vendor-neutral industry association for the promotion of OpenDoc founded by Apple Computer, Inc., IBM Corporation, and WordPerfect. Adobe became a sponsor of CI Labs, agreeing to provide financial support, while Lotus joined as a full member, a lower level of involvement.

Adobe immediately announced its OpenDoc development plans. Steve Warnock, Adobe's chief executive officer, said, "We think OpenDoc is a powerful way to achieve cross-platform component software. 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." Additionally, the company plans to develop OpenDoc versions of its major software packages.

Lotus did not specify whether it would support OpenDoc but said it would be evaluating OpenDoc technologies for use in its products. Lotus already uses the OpenDoc storage technology, Bento, in some of its applications.

Implications/Opinions: We're sure that many of you are feeling the pressure to make a decision today about whether or not you'll adopt OpenDoc or OLE (that is, Microsoft's own component software architecture). As a case in point, take the recent *MacWEEK* headline, "OpenDoc, OLE Squaring Off." Apple is obviously touting the addition of Adobe and Lotus to the CI Labs fold as a strong endorsement of OpenDoc, and Microsoft . . . well, being an Apple publication we're not going to give any ink to their marketing efforts.

We have a simple answer for you: By marching down the OpenDoc path first, you don't have to make that decision. Support the OpenDoc architecture, and you can reach both the OpenDoc *and* OLE markets. See Gregg Williams's Strategy Mosaic on page 1 to find out more about the strengths of that approach.

Intel Muddying Its Own Waters?

Intel is starting to hedge its public commitments about how long the 80x86 architecture will last and what will come after it. According to a report in *Electronic Engineering Times,* Gordon Casey, Intel's investor relations manager, said P7 (the follow-on to the P6 chip due in 1995) is "not defined at this point. The P6 will roll out this year as another generation. Beyond that, how we phase in the new architecture is something we have under study."

The "new architecture" is Intel's joint-venture chip with Hewlett-Packard, and is generally referred to as *HIP.* In the past, Intel has carefully avoided calling HIP a new architecture, in an effort to convince customers that 80x86 is not obsolete.

Electronic Engineering Times continued: "In any case, the traditional X86 architecture appears to be stopping at 32 bits with the P6. Casey told an investors conference two weeks ago, in a meeting closed to the press, that 'we knew that when we moved to 64 bits, it would have to be another architecture.'"

Implications/Opinions: By committing to new chips, changing that commitment, promising a new architecture, and then swearing it's not a new architecture, Intel is creating uncertainty about its future. We think this will eventually cause PC customers to seriously reconsider their purchase plans. In Intel's own words, "It's time to stop and ask directions." Apple's aggressive strategy to license the Macintosh operating system and its successful efforts to provide great DOS and Windows compatibility with Power Macintosh systems are designed to make these users switch to the Macintosh platform. What this means to you is an increasing market for "native" PowerPC versions of your software.

A Window of Opportunity for Macintosh Products

Reportedly IBM has been trying to persuade Microsoft to develop a PowerPC port of Chicago—recently given the official product name *Windows 95.* In response, Brad Chase, manager of Microsoft's Personal Operating Systems, said that not only has Chicago always been an 80x86-only operating system, but that "it always will be." Microsoft maintains that porting the 80x86-dependent Chicago to another platform will simply take too long.

Implications/Opinions: This means that the Macintosh operating system will be the only mainstream desktop operating system available on RISC for the foreseeable future. This should make the Macintosh both an appealing platform to potential Macintosh clone licensees as well as an attractive platform for PC users who want to move from CISC to RISC but are reluctant to adopt a heavyweight operating system like UNIX® or Windows NT.

We can't say it strongly enough: The time is now to develop and market Power Macintosh products; Apple and the Macintosh development community—that means you—have a unique window of opportunity to establish a stronghold in the market with next-generation RISC personal computing technology.

Multimedia-Hungry Marketplace

Projections of CD-ROM sales are increasing greatly based on strong sales; the latest Dataquest estimates are that 1994 shipments will be 17 million units. At the same time, the trade press is increasingly taking note of problems with PC multimedia; in its September cover story on

Apple Directions Online—December

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

"plug and play," for instance, *BYTE* magazine noted the high rate of returns of PC multimedia products.

Implications/Opinions: Apple's aggressive push of CD-ROM drives into the marketplace positions Macintosh multimedia title developers to benefit from increased popularity of the technology. If you have Macintosh products in the multimedia market, you need to clearly state the huge Macintosh ease-of-use advantage customers have when they run your products. There are two reasons you'll want to join Apple in doing this: first, this will help maximize sales of your Macintosh titles to multimedia-hungry customers. Second, you'll want to distinguish your Macintosh titles so that they're not confused with DOS/Windows products, which may be increasingly hurt by negative publicity about the PC side's problems in multimedia. ♣

Strategy Mosaic

OpenDoc

continued from page 1

from the figure "Cross-platform software made easy" on page 5, both versions have the same features and use the human interface of their native platforms. Not only that, each version's documents could be manipulated, without modification, by the version on the other platform, thus achieving cross-platform document compatibility with absolutely no extra effort.

For details, read on—but first, an overview of OpenDoc.

What Is OpenDoc?

Remember the home-entertainment consoles of the 1950s and 1960s, which housed a turntable, radio, and sometimes even a television set in one large piece of furniture? In the end, they didn't work too well—they were bulky, you could use only one component at a time, and when a component stopped working, you had to send the entire thing to the repair shop—and you needed a pickup truck to do it.

Today's overweight, feature-heavy applications are the home-entertainment consoles of the software world, and OpenDoc parts are like the plethora of consumer-electronics devices we have today. Component home-entertainment devices made the consumer-electronics market explode and gave customers a wide variety of choices—not just products to choose from, but also lifestyle choices. (Where are the televisions, radios, and CD players in your house? Not in one place, I'll bet.)

OpenDoc parts are smaller software packages that do one thing—calculations, text processing, graphics—well. Click on a spreadsheet in a document, for example, and the spreadsheet "part editor" handles the user's interaction with the spreadsheet data. Click on some text, and the text part editor takes over. "Part viewers" can only display and print the contents of a part. By selling a part editor and giving the corresponding part viewer away free, you can ensure that anyone can view and print an OpenDoc document, while still maintaining a market for your product.

Advantages of Component Software

Creating your software solutions as one or more parts has numerous advantages:

- An individual part does only one thing, so it's simpler to write, debug, and maintain.
- Your time to market is shorter and your development cost is lower.
- You no longer have to Do It All to make a commercially viable product. If you want to write a state-of-the-art graphing program, you don't also have to write and maintain a spreadsheet and a text processor—you just have to do what you do best.
- If you want to upgrade your software, you need only upgrade the relevant parts. You don't have to retest the entire application, nor do you have to worry about unexpected bugs in, for example, your file-system code when you upgrade your spelling-checker code.
- You can correct bugs more quickly and distribute the new versions to your customers more cheaply.
- You can add new features by adding individual parts, or you can reach new markets by tailoring your parts to each market's needs. In both cases, the overhead is considerably less than that of creating and maintaining multiple versions of a monolithic application.

In addition, Apple believes that the OpenDoc architecture will generate additional benefits:

- OpenDoc will foster innovation, by making it easier and less expensive to bring new products to market.
- It is a platform-neutral, "open" system, meant to develop into an industry-wide standard. OpenDoc gives companies access to the technology and allows them to participate in its evolution. In fact, anyone who joins

OpenDoc Timetable

I hope you've been reading the OpenDoc documentation Apple Computer, Inc., has made available (hint: it's on the *1994 WWDC Technologies* CD), because the real thing is coming at you soon. Apple expects to ship a late beta version of OpenDoc on CD-ROM by the end of the year; this CD will go to all Apple Associates, Partners, and recipients of the monthly Apple Technical Mailing, as well as to lots of other people. Apple also expects to ship an alpha version of the OpenDoc Parts Framework in early 1995. Further out, Apple expects to ship the first commercial release of OpenDoc for the Mac OS late in the first quarter of 1995.

OpenDoc is progressing on other platforms. WordPerfect, the Novell Applications Group has implemented OpenDoc for Windows. According to a company representative, OpenDoc for Windows is currently in an alpha version, and you can request a copy by sending Internet mail to opendoc@wordperfect.com. OpenDoc for Windows currently lags behind the Macintosh version by four to six weeks, which puts OpenDoc for Windows at beta in early 1995 and the first commercial release in the second quarter of 1995.

IBM is working on OpenDoc for OS/2 and has publicly shown OpenDoc parts working in the OS/2 environment. An IBM representative told me that IBM expects to have beta versions of OpenDoc for OS/2 in early 1995, with the first commercial version shipping in the first half of 1995.

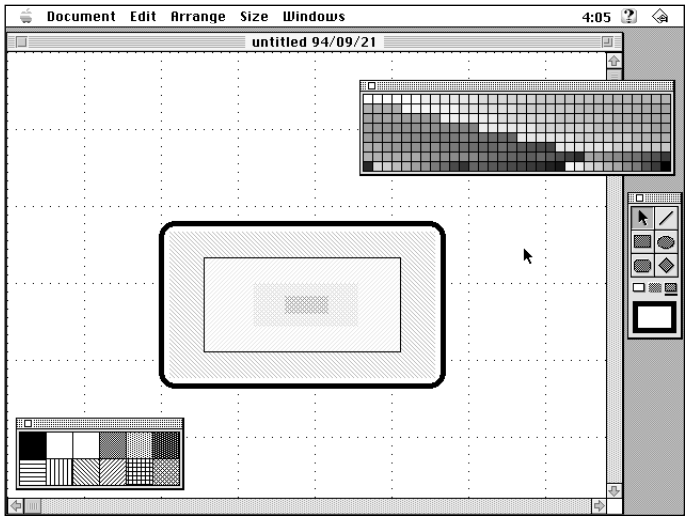
CI Labs can get the complete source code for OpenDoc.

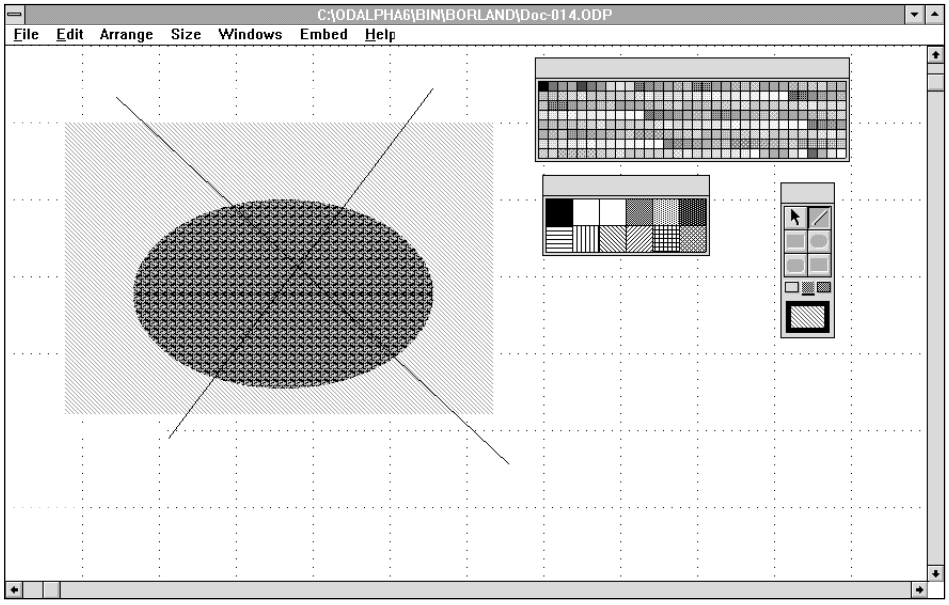
- It will "level the playing field," providing a software development environment in which smaller companies will have a better chance to succeed than they do today.
- It will generate new product opportunities and new categories of software.
- It will make software simpler and allow users to be more creative. This, in turn, will encourage users to buy more software, and it will encourage people who currently don't use computers to begin doing so. The result is a larger audience for all OpenDoc software.
- It will allow in-house developers, consultants, system integrators, and value-added resellers to more easily provide customers with better solutions, across multiple platforms.

A Framework for OpenDoc

Apple conceived of OpenDoc as a way of harmonizing many of the conflicting priorities just mentioned. OpenDoc is a good foundation for meeting these priorities, but it doesn't offer a full solution:

- To maintain maximum cross-platform flexibility, OpenDoc doesn't include a platform-independent imaging model. This means you'll have to rewrite your imaging code for each new platform you want to support. (Granted, you may want or need that *n*th degree of control, but in many cases, the same basic code can be used in multiple situations.)
- OpenDoc also doesn't handle a number of standard behaviors, including standard application behavior (for example, initialization, idle-time tasks,





Cross-platform software made easy. The OpenDoc drawing part (top) and OLE 2.0 drawing object (bottom) were both created using the OpenDoc Parts Framework and essentially the same source code—only 22 out of 8,000 lines of code had to be changed to port the original Macintosh version to Windows.

communication with other applications) and parts of the user interface (menus, controls). OpenDoc *shouldn't* implement all these things, because doing so would limit its adaptability to future platforms. Still, if someone were to write such code once, it could probably be used over and over again.

Can you say *framework,* boys and girls?

OpenDoc Parts Framework

The OpenDoc Parts Framework (OPF) is to component software what MacApp is to monolithic applications, plus you can use it to create both Mac OS and Windows software with one development effort. Like MacApp, the OpenDoc Parts Framework is written in C++ and contains numerous classes that implement useful functionality. And since the OpenDoc Parts Framework is written using object-oriented programming techniques, you can often get your work done by creating subclasses of existing classes, instead of writing procedural code from scratch. Indeed, the combination of object-oriented programming and a software framework offers one of the highest jumps in programmer productivity known today.

The example given at the beginning of this article—the 8,000-line OpenDoc part that an engineer translated to the Windows platform by changing 22 lines of code—was created with an early alpha version of the OpenDoc Parts Framework. Granted, 99.7 percent code reuse is unrealistic in most real-world situations, but this example proves that such a level of reuse is possible. Apple engineers hope to see at least 80 percent code reuse when building OpenDoc parts using the OpenDoc Parts Framework.

See the text box "OpenDoc Parts Framework Features" to see a list of the major features that OPF takes care of for you. In addition, here are some further goals of the OpenDoc Parts Framework:

- portability across multiple platforms (Mac OS and Windows first)
- usability for software other than OpenDoc parts
- modularity through components and layers (more on this later)
- the ability to add new functionality normally—that is, through the platform's native application programming interfaces (APIs)
- support for popular development environments/compilers on both the Mac OS and Windows platforms
- support for Microsoft OLE (through OpenDoc)

OpenDoc parts have many standard behaviors that, normally, you will have to program yourself. The OpenDoc Parts Framework offers the significant advantage of giving you carefully written code that implements these behaviors correctly—which means less work to make your part conform to the OpenDoc standard and more confidence in the correctness and reliability of your part.

Components and Layers

Apple has been working with software frameworks for many years, and Apple engineers (and third-party programmers) have learned a lot—primarily from using and improving Apple's framework for monolithic applications, MacApp. One of MacApp's weaknesses, however, is the interconnectedness of its object classes. So many classes depend on each other that it is impossible to use a subset of MacApp—you have to compile all of it (with subsequent performance and code-size problems), even if you don't use large subsections of it. Furthermore, MacApp can only be used to build stand-alone applications. The OpenDoc Parts Framework uses the concepts of components and layers to achieve greater modularity and the ability to be used for software other than OpenDoc parts.

An OpenDoc *component* is a collection of classes, definitions, and utility functions that work with each other to provide a specific service to the framework. Examples of components include a string component, a platform-independent graphics component, and an embedding-part component (one that can embed other parts within itself).

Components are grouped into three different *layers,* depending on the complexity of the components and their interdependencies. In particular, a component may depend upon other components in the same layer or a layer below it, but not on a component in a layer above it. In addition, component dependencies must go in one direction; that is, if component A depends on component B, then B must not

OpenDoc in a Windows world. This screen shot shows, among other things, that OpenDoc parts can be used in any Windows application that can interact with OLE objects.

depend on A. These design constraints make it possible to use only a subsection of the OpenDoc Parts Framework to create non-part software that is platform-independent.

Foundation Layer

The lowest layer of OpenDoc is called the *foundation layer;* its components are largely platform-independent, and they do not require OpenDoc to work. This means that you can use the foundation components to build non-part software and still end up with source code that is highly portable. The foundation layer contains nine components:

- common component (generic type definitions and functions used by the rest of the OPF components)
- task globals component (simplifies writing code that uses global data)
- debug component
- run-time type identification component (provides a mechanism for run-time identification and checked type casts)
- utilities component (miscellaneous components)
- collections component (efficient, domain-independent data structures and algorithms)
- streams component (provides input and output services)
- archivers component (saves and restores dynamic objects in their current states)
- strings component (classes for manipulating string objects, using whatever character encoding is used on the current platform)

OS Layer

The *OS layer* provides a platform-independent interface to common system services such as graphics and memory management. These components require the presence of OpenDoc, but you can still use them to build non-part software. This layer includes the following components:

- memory component (handles memory allocation and movement)
- files component (provides a platform-independent way of working with files)
- resource component (provides a platform-independent way of working with resources)
- OS streams component (specializes the streams component to the operating system in use)
- menu component (provides a platform-independent way of working with menus, including pull-down hierarchical menus)
- event component (provides a platform-independent way of working with operating-system events)
- graphic component (provides a platform-independent graphics model that uses QuickDraw on the Macintosh platform and GDI (Graphics Device Interface) on the Windows platform; the abstract model creates graphics using the elements of shapes, inks, styles, and transforms)
- OS miscellaneous component (contains miscellaneous classes, including cross-platform alerts, cursors, and standard file dialog boxes)

Parts Layer

The *parts layer* uses the components in the two layers below it to create cross-platform OpenDoc part editors. In essence, the "framework" part of OPF is in this layer. It has one main component, a part component, which allows the creation of part editors that have content, that can be embedded in other containers, and that can have parts embedded in them. These components include OpenDoc's support for drag and drop, linking, scripting, window scrolling, pane-splitting, and undo operations.

So What About the Windows Market?

OpenDoc is meant to be implemented on multiple platforms, and WordPerfect, the Novell Applications Group, is implementing OpenDoc for Microsoft Windows. So you can have OpenDoc part editors and viewers on a Windows system and use them to edit, view, and print OpenDoc documents, just as you would on the Macintosh platform.

But that's just the beginning: Apple's and WordPerfect's strategy ensures that you no longer have to gamble as to which component-software architecture will win—OpenDoc will support both, equally well.

WordPerfect is implementing an OLE "part-wrapper" object that makes an OpenDoc part conform to the OLE 2.0 APIs, thus allowing it to be used by any Windows application that accepts OLE objects. WordPerfect is also implementing an "OLE-container" part that allows an OLE object to be embedded in an OpenDoc part or container. (All this is possible because the functionality of OpenDoc is a superset of OLE's.) This means that OpenDoc for Windows parts will work in any Windows application that accepts OLE (both versions 1.0 and 2.0), and OLE objects will work in OpenDoc for Windows documents! In addition, by porting WordPerfect's "part-wrapper" code back to the Macintosh platform, Macintosh OpenDoc parts will work correctly in the Macintosh applications that support OLE but not OpenDoc.

Furthermore, since Microsoft has invested heavily in promoting OLE 2.0 to Windows developers, it's unlikely that Microsoft will change OLE in a way that would break existing OLE objects. So you can stop worrying about which component-software architecture to support–OpenDoc supports them all. (And in the unlikely event that a new version of OLE were to break OLE 2.0, WordPerfect could revise its OLE wrapper object, and existing OpenDoc parts would continue to work under the new version of OLE—even though existing OLE 2.0 parts would have to be modified.)

The Windows screen shot labeled "OpenDoc in a Windows world" (page 6) shows the end result of OpenDoc's compatibility with OLE. The large rectangular object filling up most of the Excel spreadsheet is an OpenDoc part. Inside it are an OpenDoc text part, a graphics part, an OpenDoc "clock" part, and a Microsoft Word document embedded as an OLE object. Note that the OpenDoc clock part is not rectangular (OLE objects are limited to rectangular shapes) and that the Word OLE object is inside the OpenDoc part, which itself is inside Microsoft Excel, an application that accepts only OLE objects.

This "OpenDoc everywhere" approach has several important advantages:

- You can create software for both the Macintosh and Windows platforms with considerably less than twice the effort. (And the OpenDoc Parts Framework can reduce your work even more.)
- You can reach the Windows market without having to learn OLE. Programmers at a recent OpenDoc "kitchen" (see the news item on page 9) reported struggling with OLE for months to little effect and, in contrast, getting OpenDoc parts working in less than a week. Since OpenDoc parts work with all OLE-enabled Windows applications, why put yourself through all that grief (not to mention greatly decreased productivity)?
- OpenDoc offers you an easier development effort, a newer technology that translates to more opportunities for innovative products, a longer life cycle

for your source code, and a larger audience than OLE can provide.

• Because of OpenDoc compatibility with OLE, developing for OLE and OpenDoc is not an either/or situation. You can switch over to OpenDoc now, even if you've already committed to OLE; your OpenDoc parts will work correctly with your OLE objects, and vice versa.

• Finally, there is the delicate issue of Controlling Your Destiny. OLE is a closed standard that Microsoft controls, and it is based on the needs of one platform—Windows. Microsoft keeps OLE's structure and operation proprietary, and you cannot make good implementation decisions because you don't have access to OLE's internal details. OpenDoc, in contrast, is a platform-neutral, open standard that is supported by multiple companies under the auspices of an independent organization, Component Integration Laboratories (CI Labs). Anyone who becomes a member can influence the evolution of the standard and get full source code to OpenDoc.

OpenDoc and Your Future

Committing to component software, you may say, is a big risk—and you're right, there is risk. But with Apple, IBM, Microsoft, and other companies behind component software, it will almost certainly happen in some form—so the question becomes not *whether* to adopt component software, but *whose.*

Frankly, OpenDoc represents a better business proposition. It has a superior architecture, it's clean-slate new technology, and it allows you to create software for the Macintosh and Windows platforms from the same development effort. OLE is motivated by the desire to increase the lifespan of an older technology—that of monolithic applications. But bolting yet more functions onto an old software model is not a good solution for the developer or the customer, and it gives the most benefit to the companies with the heaviest investment in monolithic applications. True, it may make next-quarter-bottom-line sense to stay with the old technology for the short term—but that decision just delays a much higher cost further down the road.

As always, your next move depends on both your circumstances and your goals, and only you can decide what's best for your company. Are you beginning an entirely new product, or are you doing a major (or minor) revision of an existing one? Are you aiming for just one platform (either Macintosh or Windows), or are you committed to versions for both? Are you already using OLE? The answers to these questions will influence your decision, but here are some recommendations that you should consider:

• No matter who you are, it can't hurt to get familiar with "plain vanilla" OpenDoc—and you'll be able to do that by the end of the year, when Apple releases the OpenDoc beta CD. See the article "OpenDoc Programming Made Easy," on page 16 of this issue, for details on how to get started.

• Some of the first developers to create OpenDoc parts will achieve eminence in the emerging component-software market. If this is your goal, you will want to start developing OpenDoc parts immediately. The amount of work involved depends on your situation. By itself, OpenDoc takes care of many details. Probably the most important thing it doesn't take care of is imaging—which means that if you write an OpenDoc part for the Macintosh, you will have to rewrite all your imaging code when you port it to Windows. However, a good number of developers have already created their own cross-platform imaging routines. By using these routines on both platforms, they will be able to port their parts to Windows much faster than developers who must rewrite their imaging code from scratch.

• If you are starting a project from scratch (that is, you haven't started coding yet) and you are committed to delivering it on both the Macintosh and Windows platforms, you should plan on implementing it using the OpenDoc Parts Framework. Start by reading the OpenDoc and OpenDoc Parts Framework documentation on the *1994 WWDC Technologies* CD. (If you want to investigate OLE, several books exist for doing so; I've been reading the 977-page *Inside OLE 2,* by Craig Brockschmidt, published by Microsoft Press.) You can start doing serious work with the OpenDoc Parts Framework when the alpha OPF CD comes out (in early 1995).

• If you're already using OLE, investigate the advantages of OpenDoc. Start building OpenDoc parts now. In the second quarter of 1995, port them to the Windows environment and see how well they work with OLE. Once you become satisfied that OpenDoc parts are a better solution for both the Windows and Macintosh markets, do your future component-software development using OpenDoc.

Component software is virtually inevitable, and a number of companies (including Apple) believe that OpenDoc is the best technology for implementing it. OpenDoc has a strong future ahead of it, and the transition path from existing technology to OpenDoc is practical. In addition, the OpenDoc Parts Framework will give you a significant productivity boost in creating software for either the Macintosh or Windows platforms, or both. Those who adopt these technologies will reap considerable benefits. ♣

OpenDoc Parts Framework Features

Here are some of the behaviors that the OpenDoc Parts Framework handles automatically (or assists with) on both the Macintosh and Windows platforms:

• initialization, quit, idle-time processing, messaging with other applications, copy/paste, drag and drop, menu bar reconfiguration when a new application is activated

• management of graphic objects: pens, brushes, bitmaps, fonts, regions, palettes

• document and view management: storage (to a file), display, multiple views of a document, windows with split panes

• printing: device independence, user interface, print preview

• dialogs: modal and modeless

• controls: initialization, static text, buttons, list boxes, scroll bars

• dynamic type checking

• persistence: ability of any object to save its state to a persistent storage medium (such as a disk)

• collection classes (lists, dynamic expandable arrays) to support standard data structures efficiently

• string manipulation support

• memory management

• file manipulation

• time and date support

• diagnostic and debugging facilities

Apple News

Mac OS

continued from page 1

1995. Licensing will mean not only expanded business for the Macintosh operating system, but also for your Macintosh products, as well.

"Our number-one licensing objective is to create a business model that's attractive, sustainable, and profitable for our licensees, and that creates new opportunities for ISVs [independent software vendors] and other third parties," said Don Strickland, Apple's vice president in charge of licensing activities. "We believe that licensing offers software vendors a broader Macintosh platform installed base on which they can develop and provide new technical innovations."

Licensing is an important part of Apple's overall strategy aimed at significantly increasing the Macintosh share of the personal computer market. Apple's strategy also includes extending the Macintosh platform's price/performance leadership, increasing cross-platform compatibility, and expanding its worldwide marketing investments.

Apple intends to give licensees flexibility and choices regarding when and how they market their Macintosh-based systems. In the first phase of Apple's licensing efforts, Apple aims to license the Mac OS to companies with expertise that complements Apple's traditional strengths. Such expertise could include unique technical capability to create value-added solutions or distribution channels to reach customers in geographies where Apple has limited presence. Over time, Apple aims to license more broadly to a more diverse group of vendors.

Initially, Apple plans to license its core Macintosh operating system (the recently shipped System 7.5) and elements of its PowerPC processor–based hardware architecture as needed to preserve distinguishing features of the Macintosh computer, such as its traditional ease of use and plug-and-play capabilities. Apple will also provide engineering and development support to its licensees to assist them in bringing products to market rapidly.

Using the Mac OS Logo With Your Product

Last month, the new logo for the Macintosh operating system—now officially named *Mac OS*—made front-page news in *Apple Directions* (see "Introducing Mac OS!" on page 1 of the October 1994 issue). Apple Computer, Inc., will be using the Mac OS

logo in its packaging, advertising, and marketing materials, and it encourages you to do the same. That way, customers will easily be able to identify your products as being for the Macintosh operating system, including third-party Mac OS–compatible computers.

Apple needs your help to make the Mac OS logo visible everywhere, and has instituted a streamlined process to get you the materials you need. All you have to do is sign a no-fee license agreement and a letter certifying that your product runs with Macintosh System 7.5 and is 32-bit clean (which it is if it works on a Mac OS computer that has virtual memory turned on). Apple will process your application and send you a CD-ROM with the artwork for the Mac OS logo. All Apple asks in return is that, once you have incorporated the Mac OS logo in your product, you send Apple one copy of the complete product.

By the time you read this, all Apple Associates and Partners will have received a packet of information on this process, including the license agreement mentioned above. If you have any questions or need an information packet, call Apple Licensing at (512) 919-2645.

WordPerfect and Apple Host Cross-Platform OpenDoc "Kitchen"

WordPerfect, the Novell Applications Group and Apple Computer, Inc., recently combined their Windows and Macintosh development expertise to host the first cross-platform OpenDoc Developers Kitchen. Held in Provo, Utah, the four-day workshop attracted developers representing more than 20 companies including Adobe Systems, Inc., Claris Corporation, Oracle Corporation, and Shapeware Corporation. Using OpenDoc alpha source code, developers created OpenDoc parts from scratch and changed existing applications into fully operable OpenDoc parts in less than a week.

"OpenDoc provides a robust cross-platform architecture for creating component software," said Subhashis Mohanty of Waterloo Maple Software, a leader in advanced mathematics software. "By including development teams from Apple and WordPerfect, the kitchen provided me with a forum to ask questions regarding development on both Windows and Macintosh. Since the kitchen was limited to a small group, they were able to provide real, hands-on training."

"With OpenDoc now in the hands of more than 20,000 developers, we are starting to teach them how to make the transition to component software," said Bill Kesselring, manager of OpenDoc Technology for WordPerfect. "OpenDoc makes it very easy to convert existing code from large monolithic applications into small, lightweight components. These developers experienced this firsthand when they were able to create OpenDoc components in a matter of days."

Waterloo Maple Software

During the "kitchen," Subhashis Mohanty converted part of Waterloo Maple Software's mathematical software package, Maple V, into OpenDoc for Windows components. In less than a week, Mohanty rewrote existing code to create a part that evaluates mathematical expressions and creates numerical results in seconds.

"We found that the OpenDoc architecture was ideal for mathematics parts," said Mohanty. "OpenDoc provides us with a mechanism for building robust component software very rapidly. Waterloo Maple Software is seriously considering developing commercial OpenDoc parts for Windows, Macintosh, and UNIX."

WordPerfect is developing OpenDoc for Windows and OLEO (Open Linking and Embedding of Objects); OLEO will enable full compatibility between OpenDoc and Microsoft's OLE 2.0. Developers said that the kitchen assured them that OpenDoc was the fastest and most robust way to create OLE 2.0 objects and OpenDoc parts and to turn existing applications into components.

Environmental Systems Research Institute

"Component software will significantly reduce development costs and increase user benefits," said Keith Ludwig, a developer at Environmental Systems Research Institute, Inc. (ESRI) who attended the kitchen. ESRI develops geographic information system (GIS) software for Macintosh, Windows, and UNIX workstation platforms. "Our desktop mapping package, ArcView, is a prime candidate for conversion to component software. Today we spend considerable time developing peripheral functionality, such as charts and graphs that allow users to view data in alternate ways within ArcView. OpenDoc would enable us to focus on the areas where we can add the greatest value, developing GIS and mapping components. Our components would then work with complementary products by vendors who are following similar strategies."

Metrowerks

"Metrowerks is committed to making CodeWarrior the best programming environment for developing OpenDoc applications on the Macintosh and PowerPC," said Greg Dow, a Metrowerks, Inc., developer who attended the kitchen. "By using our PowerPlant class library included with Metrowerks CodeWarrior, developers will be able to create both stand-alone applications and OpenDoc parts using the same source code—no rewriting is required."

Future Developers Kitchens

WordPerfect and Apple are planning to co-host the next OpenDoc for Windows and Macintosh Developers Kitchen in early December. In November, Apple's European Technology Marketing group will host a three-day European OpenDoc Kitchen during the European Developer Forum in Sweden. IBM plans to host an OpenDoc for OS/2 Developers Kitchen later this fall.

"The OpenDoc development effort is unique," said Wendy Tajima, OpenDoc product line manager at Apple. "Because OpenDoc is supported by multiple companies with expertise in specific areas, we are able to team up and work together to provide solutions for cross-platform developers. Apple is looking forward to co-hosting future developers kitchens and conferences with IBM and WordPerfect."

Component Integration Laboratories

WordPerfect and Apple are two of the sponsors of CI Labs, Inc., a vendor-neutral industry association promoting the OpenDoc component software technology. (The others are IBM and Adobe.) CI Labs adopts, maintains, licenses, and supports OpenDoc's essential technologies to ensure compatibility among OpenDoc parts. CI Labs will provide reference source code, technical documentation, example software and validation services—openly, without nondisclosure requirements.

Apple, IBM, and WordPerfect, the Novell Applications Group, have all licensed major technologies to CI Labs. They are also developing OpenDoc for Macintosh, OS/2, and Windows platforms, respectively. For more information on OpenDoc, contact Andy Poupart, vice president of CI Labs, at (415) 750-8352 or Internet mail address cilabs@cil.org.

New Macintosh vs. Windows Video Available

Apple Computer, Inc., has recently updated its "Macintosh vs. Windows" video. (See page 16 of the April *Apple Directions* for our original announcement.) The new

version includes a variety of recent information, such as System 7.5 features and a benchmark study conducted by Ingram, who found that the Power Macintosh 8100/80 computer outperformed Dell's 100-MHz Pentium PC.

"Macintosh vs. Windows" is designed to help you understand and demonstrate why the Macintosh is better than a PC running Windows. It's not confidential, and you're free to share it with your customers. Copies of the 23-minute video are available for $8 through the *StartingLine* catalog (order number V10005). To order, call (800) 825-2145 (United States) or (303) 297-8070 (outside the United States).

New Mac OS SDK Saves You Time and Money

With the many recent extensions to the Macintosh operating system, recently named *Mac OS,* it's become increasingly difficult—and costly—to keep track of all the software developer's kits (SDKs) you need for incorporating new features into your application.

The Mac OS Software Developer's Kit, just introduced by Apple Computer, Inc., changes that. The new product gives you convenient and much less expensive access to the individual Macintosh system software SDKs released by Apple. Previously, 16 of the extensions supported by the first release of the Mac OS SDK cost more than $1700 if you purchased them separately. Now, you can obtain virtually all of Apple's Macintosh system software SDKs (more than 30) at once through a one-year subscription to the Mac OS SDK for $299. You'll also receive quarterly updates that include the software you need for taking advantage of future additions to the Mac OS.

The first release of the Mac OS SDK includes SDKs for the following extensions:

Apple Guide
Apple Open Collaboration Environment (AOCE)
Apple Remote Access
Apple Remote Access Modem
AppleScript
AppleSearch
AppleShare API
Apple Shared Library Manager
AppleTalk Wide Area
ColorSync
Communications Toolbox
Control Strip
Designing PCI Cards & Drivers
File System Manager
Installer
Macintosh Drag and Drop
Macintosh Easy Open
MacODBC
MacOSI Connection
MacSNMP
MacTCP
MacX25
MacX.400
MIDI Management Tools
Network Software Installer
Open Transport
PlainTalk
QuickDraw GX
QuickTime
Sound Manager
Telephone Manager
Thread Manager
XTND

Because of licensing restrictions, special distribution requirements, or changes in distribution strategies, certain SDKs may not be included in the Mac OS SDK from time to time. A one-year subscription to the new product, however, will remain the best way to stay up to date on all the software you need for taking advantage of new system software features.

Typically, the Mac OS SDK will include the following key components for each individual extension:

- the system software extension itself
- programming interfaces and libraries
- sample code
- technical documentation

An annual subsription to the Mac OS SDK will include the most current version of the SDK, shipped on CD-ROM, plus the next three quarterly updates.

The Mac OS Software Developer's Kit is available immediately from APDA. For APDA ordering information, see page 28. Members of the Apple Partners program worldwide receive one subscription to the Mac OS SDK without charge. The first release is included in the November 1994 mailing to Apple Partners; to inquire about becoming an Apple Partner, send an AppleLink message to DEVSUPPORT.

New Printers Announced at Macworld Frankfurt

Late last month, at Macworld Frankfurt (Germany), Apple Computer, Inc., announced the worldwide availability of two new printers. The LaserWriter 16/600 is a new 16 pages-per-minute (ppm), 600 dots-per-inch (dpi) printer meant for office environments. The Color StyleWriter 2400 printer improves on the Color StyleWriter Pro, while remaining affordable for the home, education, and small-business markets.

For more demanding office environments, Apple has introduced the LaserWriter 16/600 PS, a high-performance, networked PostScript™ laser printer. An optional fax card adds high-quality desktop fax capability. The LaserWriter 16/600 PS printer provides impressively sharp print quality with 600-dpi resolution enhanced with FinePrint technology for precise, smooth text and line art. It also includes optional support of PhotoGrade for grayscale image enhancement. The LaserWriter 16/600 PS can handle input from Macintosh, Power Macintosh, Microsoft Windows, DOS, and UNIX computer systems, with built-in support for AppleTalk, Novell NetWare, EtherTalk, and TCP/IP Ethernet networks. Since all ports stay active simultaneously, the LaserWriter 16/600 PS can accept jobs from a variety of computers without the user having to change a single setting. The LaserWriter 16/600 PS has a U.S. Apple price of $2,429.

The Color StyleWriter 2400 delivers crisp text and vibrant colors at a U.S. Apple price of $525. It is ideal for families, educators, and small businesses who want professional-looking documents without a steep price. The Color StyleWriter package is ready for printing out of the box, with 64 TrueType fonts, cable, inks, and integrated ColorSync software support that delivers quality color matching and superior print quality at 360 dpi. It prints up to three pages per minute with the black cartridge and one-third of a page per minute in color.

The Color StyleWriter 2400 is designed for use with any Apple Macintosh color-compatible personal computer that has 4 MB of RAM, a hard disk drive, and system software version 7.0 or later. It ships with a multipurpose tray that accommodates up to 100 sheets of paper or 15 envelopes. ♣

Technology

CD Highlights

Tool Chest Edition, November 1994

This month's disc features a subject index (roughly corresponding to the subject areas of the *New Inside Macintosh* series) similar to that introduced on September's Reference Library CD. Within each subject folder you'll find aliases to all material on the disc related to that subject. Feedback, as always, is welcome; send your comments to AppleLink address DEV.CD or to Internet address dev.cd@applelink.apple.com.

In addition to updates to the Developer Notes, Tech Notes, and Newton sample code, here are this month's new and revised packages.

AE Suite • Finder

The Finder suite contains definitions of the Apple events and other Apple-event constructs understood by the Finder. This document supersedes the information provided in Chapter 8 of the Apple event registry. The old Finder suite was developed before the Apple Event Manager and the Apple Event Object Support Library were complete. As a result, the old Finder suite did not support Apple event objects or the core Apple events. The events defined in the old Finder suite continue to be provided for backward compatibility with existing applications; the event class of events defined in the old suite is kAEFinderEvents. Because the old Finder suite does not take advantage of the current capabilities of Apple events, applications that wish to control the Finder should always use the new event suite described in this document.

Apple Bug Reporter 1.6

The Apple Bug Reporter is a HyperCard stack that allows developers to report bugs in Macintosh software and hardware. When you have completed a bug report, the stack puts it in your AppleLink Out Basket. When you next log on to AppleLink, your report is sent to APPLE.BUGS.

Version 1.6 of the Apple Bug Reporter updates configuration information (it includes new Macintosh and monitor models), adds some components, and contains some minor interface enhancements. Of course, it also contains some bug fixes.

Tool Chest Edition

Apple Guide Authoring Kit

This kit helps you create guide files for use with Apple Guide and System 7.5. (Please read the license agreement before using any of the items in the kit.)

Apple Guide marks a leap from the passive assistance of typical help systems to online active assistance that lets users accomplish tasks as they learn how to do them. Rather than read through a manual or browse through an application's built-in "help" (typically little more than a manual on disk), users can call on Apple Guide and walk through the solution to their problems simply by following a series of on-screen prompts. It's almost like having a local expert look over your shoulder and guide you through the task at hand.

With the Guide Maker 1.0 authoring tool provided in this folder, you can write your own guides to walk a user through the applications they already have on their Macintosh computer. First, you write your content in the word processor of your choice and use Guide Maker to compile your content into an Apple Guide file. Then, you simply drop the guide file into the folder of the application it applies to, and—voilà!—you've just brought Apple Guide's amazing abilities to bear on your users' most frequently asked questions.

Also included in this folder is full online documentation for Guide Maker 1.0 and the Guide Script command language, as well as convenient guide authoring aides such as Coach Mark Assistant, a tool that makes specifying coach marks (circles or other pointers) as easy as drawing them on screen.

AppleSearch Client SDK Overview

This package provides information on how to create or develop products that use (or are used with) AppleSearch. It contains two documents:

- AppleSearch Developer's Kit Overview
- AppleSearch Update File Format Reference

You must sign the AppleSearch Developer's License before using and distributing AppleSearch.

Inside This Section

AppleTalk and PPP

This package includes the draft specifications for running AppleTalk over the Point-to-Point Protocol. If you are working on this implementation of AppleTalk, you should follow these specifications as they are standardized by the Internet Engineering Task Force.

BBEdit Lite 3.0

BBEdit Lite is a freeware derivative of BBEdit 3.0, the popular and critically acclaimed text editor for programmers, online service users, and anyone else who needs to edit plain-text files.

BBEdit Lite is shipped with a QuickStart document that describes many of the application's features. Full documentation, as well as a considerable amount of additional functionality, is available with the purchase of BBEdit 3.0. See the document "BBEdit General Information" for details on obtaining BBEdit 3.0.

Note: This is *not* an Apple product. It is provided on an "as is" basis. Apple is not responsible for any problems you may encounter in its use.

Designing PCI Cards and Drivers

In the August issue of *Apple Directions,* we described Apple's new PCI bus and Open Firmware initiatives ("PCI Cards for the Second Generation of Power Macintosh," page 13). This CD folder contains an early draft of Apple's guide for developers, *Designing PCI Cards and Drivers for Power Macintosh Computers,* in DocViewer format. The guide covers the hardware and software interfaces for PCI cards and describes Apple's implementation of Open Firmware. It also defines forthcoming requirements for writing Macintosh device drivers in PowerPC code. Although Apple's PCI technology is still under development, you'll find much useful information in this prelimary document.

Dylan Related

This folder contains information and sample code pertaining to Dylan, a new object-oriented dynamic programming language (OODL) developed at Apple.

- Dylan Interim Manual: This directory contains the June 1994 interim Dylan language reference.
- Dylan FAQ 29July94: Answers to Frequently Asked Questions about Dylan. This document includes information about mailing lists and places to get sample code, implementations, and further information.
- newsletter #1, 29Jul94: A brief newsletter with some Dylan news, including the latest Frequently Asked Questions document.
- Dylan WWDC '94 brochure: Apple's two-page brochure describing the Apple Dylan environment; it was distributed at the Apple Worldwide Developers Conference in May 1994. This document was created with Common Ground, so you don't need the original fonts or applications. On Macintosh, you just double-click to view it. Viewers for Windows or UNIX are also available (contact NOHANDS@applelink.apple.com).
- MacMarlais-0.3.1.sea: This is the latest Macintosh version of Marlais, a freeware interpreter for Dylan. Version 0.3.1 still supports the old LISP-like syntax. A new version, 0.5, is available for UNIX and Windows, and the Macintosh version will be released in the next week or so.
- Marlais 0.5 (sources only): Source code (in C) for Marlais that supports the newer infix (algebraic) syntax.
- Marlais-0.5 for Windows.zip: A Windows port of Marlais that supports the newer infix (algebraic) syntax.
- Mindy-1.1 (sources only): Source code (in C) for Mindy, a freeware byte-code compiler for Dylan. This compiler supports the newer infix (algebraic) syntax; it's been compiled for several implementations of UNIX, but no version for Macintosh or Windows has been released yet.
- Thomas 1.1/Gambit 2.0: This is an experimental implementation of Dylan. Thomas is implemented in Scheme, and runs on many versions of Scheme on many platforms. In particular, we include a stand-alone Thomas interpreter for the Macintosh based on Gambit, a freeware version of Scheme for Macintosh (and 680x0 UNIX computers).

File System Manager SDK

The File System Manager provides a systematic way for one or more foreign file systems to interact with the Macintosh file system using high-level language interface.

This development kit includes the following features:

- File System Manager: A system extension that implements version 1.2 of the File System Manager and the extended Disk Initialization Package. This extension is compatible with system software versions 7.0 and later.
- FSMGlueLib.o: The glue code for the File System Manager service routines.
- FSM.h. FSM.p and FSM.a: The Universal Interface files for the File System Manager. This version of the interface files was built to work with the Universal Interfaces 2.0a1 from E.T.O. #15, MPW pre-release. However, for 680x0 development, FSM.h works perfectly with the current release of Universal Interfaces on E.T.O. #15.
- Guide - File System Manager: The documentation for the File System Manager in Apple DocViewer format.

To license the File System Manager extension for distribution, contact Apple Software Licensing.

Installer 4.0.3 SDK

Installer 4.0.3 provides the latest Apple technology for installing and upgrading software to both 680x0 and PowerPC processor–based Macintosh systems.

Version 4.0.3 is a bug-fix release of the Apple multidisk installer that also offers the new InstaCompOne decompression atom extender. The documentation has also been revised to address common questions and issues that users of the previous version have raised.

Macintosh Common Lisp Utilities

This folder includes the latest patches and interface files for users of Macintosh Common Lisp (MCL). This is a maintenance release, with some bug fixes and support for MCL on newer Macintosh platforms such as the Macintosh Quadra AV series. If you currently have MCL 2.0, the enclosed files will update you to 2.01. In addition, there are tutorials and discussions of various Lisp-related topics.

Newton Sample Code 1.2

This folder contains Newton Q&A documents (in both DocViewer and Microsoft Word formats); sample code compatible with the Newton MessagePad, Newton MessagePad 100 and 110, and Sharp ExpertPad; and several articles on Newton development. Each Newton project includes a text file

please turn to page 20

Human Interface

Microchips and Pride

By Peter Bickford

Here's a quiz for you systems analysts out there. I'll give you a real-life design problem and you'll need to brainstorm the appropriate solution. Use a #2 pencil and write clearly. You have ten minutes to complete this task.

The Situation

Your client is the head of food service for a group of hospitals in the Denver area. There is a large central hospital and a number of smaller hospitals that lie within a six-mile radius. No network connections exist between the main hospital and the outlying hospitals.

Patients in the various hospitals are given individualized menus that take into account their special dietary requirements, from which they are to check off their food service selections for the next day. A courier service runs these menus from the outlying hospitals back to the central hospital where the food is prepared; then it takes the food over to the outlying hospitals to be served. Unfortunately, late arrivals, operations, and various other conditions mean that as many as half the menus aren't filled out until an hour or so before meal time. This causes no end of chaos, as late orders must be prepared and individually trucked over to the special patients.

The food service department is not currently computerized, but other parts of the hospital use a combination of PC clones and Macintosh computers connected to an AS/400 mainframe.

You are a systems analyst who has been hired as a contractor by the food service manager. Propose a workable system that solves the menu delivery problem. Remember to show your work.

OK, time's up. Pencils down. Let's see how you did. . . .

- Give yourself *ten* points if you proposed a system whereby a network is established between the hospitals, the completed menus are passed through OCR scanners, and the information is transmitted back to the hospitals' mainframe, where it is fed into a custom program that prints menus for each tray at the central hospital.
- Add *five* bonus points if your contract specified using all-Apple equipment, taking advantage of our excellent PC connectivity solutions (a plug for true plug 'n' play).
- Pick up another *ten* points if you doubled the expected cost of the system before presenting it, then added line items for training time, an extended maintenance contract, and an upgrade agreement. Make sure the total comes to at least several hundred thousand dollars. (This is the mark of a real professional, and, incidentally, the approach that the actual systems analyst in this case took.)
- On the other hand, give yourself *1,000* points if you told the food service manager simply to get a couple of extra phone lines, put a fax machine in each hospital, and use them to send in the menus as the patients complete them.

In Technology We Trust

If you're like me, you were mentally constructing a star network between the sites before you finished the first paragraph. By the end of the story, I was wondering whether I should be using IPX or AppleTalk. It was only later that I bothered to think much about what the food service manager was really trying to accomplish—namely, moving menus from one location to another, a task that didn't require a network (or even computers) at all.

I'm sure some of us got involved with computers because we watched too many late-night TV commercials promising us that learning COBOL would bring us fame, fortune, and giddy looks from attractive members of the opposite sex. Real computerphiles have a name for such people: *management.*

The rest of us got involved with computers because we share a common belief that computers make life better. This faith sustains us through all-night debugging sessions, inexplicable system crashes, and countless hours spent wondering why the laser printer would rather commit *sepuku* than print page 3 of our document.

Our trust in computers is a wonderful thing in this cynical world of ours, but it has a downside. We tend to leap right in with technical solutions, then spend all our time refining the details—the "how" of the system. Instead, what we need to do is take the time to figure out the "why." Why do users need this system—what are they really trying to accomplish, and is this system really the best way to meet those objectives? By asking "why," we often find out that the problem we were trying so hard to solve is actually the wrong problem, as in the food service manager's situation described earlier.

The weirdest part comes when we have the audacity to simply dream up a cool technology, show it to a potential customer, and say, "Look! This is really cool! Buy it and I'm sure you'll find a use for it!" Without pointing fingers at the all-too-easy targets, we should remember that when personal computers first became available, companies tried selling them to homemakers as multithousand-dollar recipe keepers. Thank heavens somebody came up with truly useful things like word processing, or we'd all be out of a job.

User-Centered Development vs. Technology-Centered Development

Of course, we all know how product development *should* go:

1. Start by getting to know what the unmet needs of your target customers are.
2. Look beneath those needs to see the underlying problems.
3. Work cooperatively with your potential customers to brainstorm solutions to those problems.
4. Implement the solution, checking often with customers to make sure you don't wander off the track in terms of features, price, appearance, and so on.

Too often, though, developers employ a few shortcuts to "streamline" the process. For starters, they assume the users are just like themselves (only a little bit dimmer). Thus, developers accomplish step 1 by figuring out what sort of product *they* would want. Then they omit step 2 entirely in the rush to make sure their product has a longer feature list than its market competitors. Step 3 is considered too much of a hassle, since customer visits "would only slow development." This concern is understandable, since the programmers are up to their ears supporting different event suites, operating system versions, and thousands of other technical details.

The only time the customer actually gets to sneak into the process is late in the beta stage of implementation. This is the point at which the seed sites start sending back problem reports explaining why the "frozen" parts of the interface are completely unusable, and marketing discovers that the product won't really sell unless it has a completely different feature set and costs about half as much.

If they're lucky, the company exists long enough to fix the problem in version 1.1. If they're stupid, they ignore the whole thing and hope the users will come around to their way of thinking.

True Confessions

Microchips and pride are a heady mix that can lead to your downfall if you're not very careful. They make you believe technology is the answer to everything, and that if you could just build the right system, you could bring enlightenment to the great unwashed user base (who are, of course, just like you, only slightly dimmer).

At this point, I should confess that not so long ago, I fell under this delusion and nearly spent months developing a really terrible product as a result. I'm an inveterate comic book collector and had developed a fairly involved program for tracking my own collection. One day, it occurred to me that I could rig my program to print bar codes for each of my comics, which I could then scan when I sold one at a convention. I loved the idea of writing something that went "gleep!" every time I wand-scanned a comic, effortlessly tallying sales, adjusting inventories, and so on. Moreover, I figured I could come out with a special version of my program and sell it to comic shops.

Fancying myself Mr. EveryUser, I implemented several different prototypes on my own collection and ran them through trials at various comic conventions where I was selling books. Nothing ever really ran quite as smoothly as I thought it would, but I was convinced that if I could just get the kinks out, comic retailers would welcome my program as their high-tech savior. (I probably harbored some unconscious fantasy wherein the grateful retailers would line up and say, "Gee! How can I ever thank you, Mr. EveryUser?" —"Aw shucks, it was nothing!" I would reply with a winning grin. . . .)

Then I went to a convention of comic retailers and started asking them about how they ran their businesses. It was a rude awakening. For starters, only a very few did any of the basic inventory control that is necessary to support a point-of-sale system. Most were small shops with personnel so busy running the counter that retroactively counting and labeling tens of thousands of comics was an unthinkable task. Moreover, whereas most commercial retailers only have to worry about, at most, a few thousand items (each assigned a so-called SKU stock number), comic dealers often have to track as many as 100,000 SKUs to cover the myriad comic books, posters, and whatnot on their shelves. And, each month, they have to add about 700 items to that total.

As if that weren't bad enough, books regularly changed SKUs during their shelf lives, moving from uncoded new inventory to labeled back issues, then often to $1 or 50¢ clearance boxes. The sheer workload involved in managing such inventory at the book level was more than almost any shop could handle. Never mind reconciling computer systems and other chores my system would have added. For the vast majority of retailers, a point-of-sale system would have actually made their lives worse.

There's a happy ending to this story, however. After ditching my initial ideas (and code!) I sat down and asked the retailers what they were having trouble with in their shops. As it turned out, they desperately needed something to help them keep track of their customers and their regular orders, as well as to handle shop tasks such as printing title divider labels. It wasn't as glamorous as the system I had imagined, and it certainly wouldn't go "gleep!", but it would make life better for them. That's the system that eventually got developed.

Recurring Themes and Hard Solutions

Hopefully by now, you're getting a terrible sense of déjà vu (I know I am!). The point we keep coming back to is that users need to be involved at the beginning and all through a product's development. Without their feedback, your development efforts will be like taking target practice in the dark.

In writing these columns, I always feel guilty bringing up a problem unless I can point to some straightforward solution. However, this time I'm afraid I don't have any clever techniques or devices to offer. All I can do is get out my trusty developer voodoo doll and wish plague and pestilence upon the houses of companies who ignore users, and long lives and riches for companies who put technology in its proper place: serving the needs of users.

Till next time,
Doc

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

OpenDoc Programming Made Easy

By Dave Bice

Let's say you've heard how revolutionary OpenDoc is, and you'd like to start programming with it. You've heard that developers create part editors instead of conventional monolithic applications, and that users employ those part editors to create and manipulate parts, the elements of an OpenDoc document. You suppose that the development process is not the same as in conventional application programming, but you're not sure how different it is. Will it be difficult, will it be worth it, and what's the best way to start?

You'll find that development of an OpenDoc part editor is somewhat different from writing an application, but the differences are mainly in things you *don't have* to do. Writing a part editor is likely to be *less work* than writing an application. Individual part editors are typically much smaller than applications, because OpenDoc takes care of many tasks for you, and because an editor focuses on one specific task or a set of related tasks. A large application package is, in the OpenDoc world, replaced by a suite of separate part editors. It's a division of labor into modular chunks. Each chunk—each part editor—is therefore quite manageable to develop. Writing your first part editor should be fairly easy.

Development of a part editor will be worthwhile because you will be investing in Apple's (and your) future. What you learn by writing your first part editor will carry you through all future OpenDoc development, whether you create individual part editors, large packages of integrated part editors, or part services (OpenDoc components that provide specialized functions for parts and documents), or even if you simply convert your existing applications to embed OpenDoc parts. Writing a part editor also has these extra advantages:

- OpenDoc's platform-neutral and language-neutral object model (see the text box "The Object Underpinnings of OpenDoc" on page 18) ensures that your part editor will have maximum leverage across the marketplace, interacting amicably with part editors that were compiled with other compilers and written in other languages.
- Your parts will be fully compatible with objects created for Microsoft's Object Linking and Embedding (OLE 2.0) technology. (See the text box "OpenDoc-OLE Interoperability" on page 17.)
- OpenDoc's application programming interface (API) is economical and efficient. Writing a completely functional part editor requires overriding only about two dozen methods of the class ODPart. The maximum number of ODPart methods you can override is 62.

OpenDoc is still under development, but it is quite stable and early versions are already in developers' hands. Right now, the best way to start programming for it is to examine the existing samples (at least two fully functional part editors will ship with the beta SDK), read the existing documentation, and then use Apple's PartMaker application to create a part-editor shell that you fill in with the content particular to your part editor. Soon you also will be able to use the OpenDoc Parts Framework (OPF) to make part-editor development even faster and easier. (See this month's Strategy Mosaic, "OpenDoc *Is* Cross-Platform," for more on OPF.) And in the future you will also be able to take advantage of the Container-Application Library (CALib), to more easily convert your conventional applications to OpenDoc-savvy container applications, into which users can embed all kinds of OpenDoc parts.

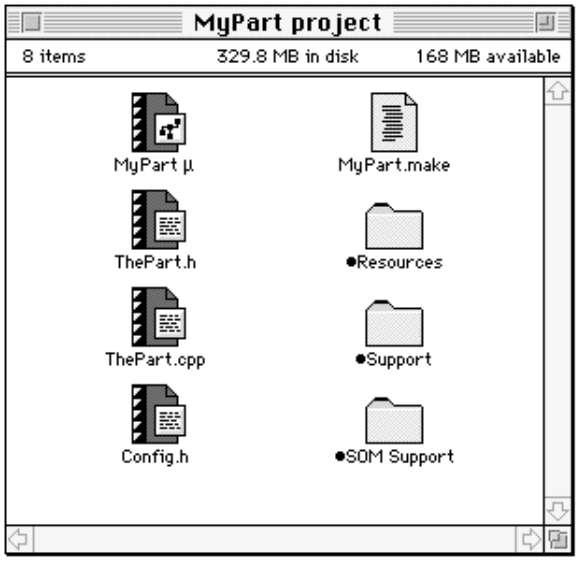
Start With PartMaker

The OpenDoc for Macintosh Beta SDK (due by the end of the year) includes a folder called *PartMaker* within the Development folder. (The exact layout of the CD has not been determined as of the time of this writing.) In the PartMaker folder you'll find the PartMaker application plus one or more PartMaker documents to use with it.

PartMaker was created by Tantek Çelik of the OpenDoc development team and Eric Soldan of Apple Computer's Developer Support Center, as a convenience for developers new to OpenDoc. You can use the PartMaker application and its documents to create the skeleton of a project, with all the files you'll need—including a makefile or project file—to compile a simple part editor.

Running PartMaker is very simple. Suppose, for example, that you want to create a project for developing a part editor in C++ using the Metrowerks CodeWarrior compiler:

1. Drag the appropriate PartMaker document (it will be named something like *C++ leaf editor*) and drop it on the PartMaker application. PartMaker launches and presents a dialog box. Alternatively, double-click on the PartMaker application icon; you will first be asked to select a PartMaker document, and then the dialog box will appear.
2. Enter the class name of your part editor (that is, the name of your subclass of ODPart, such as "MyPart"), and click the Create Part button.



Typical PartMaker output.

3. In the standard file dialog box that appears next, enter the name of the folder into which the source files should be placed (such as "MyPart Project"). Click the Save button.

Each PartMaker document is like a compressed set of files and folders, in the proper configuration for development with a given language and compiler. (At the time of this writing, it is not yet known how many different PartMaker documents will be included on the CD.) When PartMaker processes the document you drop on it, it expands the document into the needed set of files and folders, all properly named and located according to your specifications, and it also inserts the proper editor name where needed throughout the skeleton source code and resource definitions that it creates.

For C++ development, PartMaker creates a set of files (such as those listed in the "PartMaker Output" text box on page 19) in your designated project folder, which you can use for development in either MPW or CodeWarrior. The screen shot "Typical PartMaker output" (Page 16) shows the icons for the files and folders created.

Build Your Part Editor

Now that the initial housekeeping drudgery associated with developing a part editor has been done for you, you can immediately compile and link the output of PartMaker without error, using the generated project file (for CodeWarrior) or makefile (for MPW). Before you do any more coding, you'll probably want to see what your part editor can do (it probably just writes "Hello" within its frame). To do that, you first need to create some stationery.

Create Stationery

Remember that a part editor by itself is not a part. Users create parts, and parts contain data of some sort, stored separately from their editors. When the user manipulates your kind of part in a document, it is the code of your part editor that helps make those actions possible, but the user need not be aware of that. Users don't launch part editors directly; they just manipulate parts.

Therefore, to make the part editor that you have just created accessible to the user, you need to create its very first part—preferably a stationery document. You can create a stationery document by dragging and dropping your compiled and linked part editor onto the OpenDoc extension itself.

Once you have created the stationery, you can then become the user of your own part editor. You either double-click on the stationery you have created to launch a separate OpenDoc document consisting of your part alone, or you drag your stationery into an open OpenDoc document, to embed your part within it. You can, of course, embed or launch multiple separate parts, all manipulated by your one part editor.

Congratulations! You have successfully written and used your first part editor.

Add Features

Once you have created your initial part editor, you can add features one by one, to give it more capability and to learn more about how OpenDoc works.

To make your part editor capable of performing tasks that are actually useful, you will need to fill in the implementations of your methods. Your part editor needs to implement only those methods necessary for its specific functionality; the PartMaker sample already has several important methods, such as InitPart, HandleEvent, and Draw, filled in enough for its simple needs. (Note also that you won't have to worry about System Object Model [SOM] calling conventions or directly subclassing ODPart when you write your methods; the SOM object defined in the SOM Support folder takes care of that.)

Use the comments in the .cpp file, examine other sample code,

OpenDoc-OLE Interoperability

Microsoft's Object Linking and Embedding (OLE) technology is a compound-document architecture with features similar to some of those of OpenDoc. OLE 2.0 is currently available on both the Windows and Macintosh platforms.

OLE 2.0 has a number of significant differences from OpenDoc, however, including selection models that require activating entire applications rather than the relatively lighter-weight, more efficient OpenDoc parts. Unlike OpenDoc, OLE 2.0 does not support irregularly shaped part frames, nor does its scripting system include standard scripting events. OLE 2.0 is not an open standard, in that its source code is not available and there is no multivendor consortium that controls its distribution and evolution. (Component Integration Laboratories—CI Labs—is the consortium that handles those responsibilities for OpenDoc.)

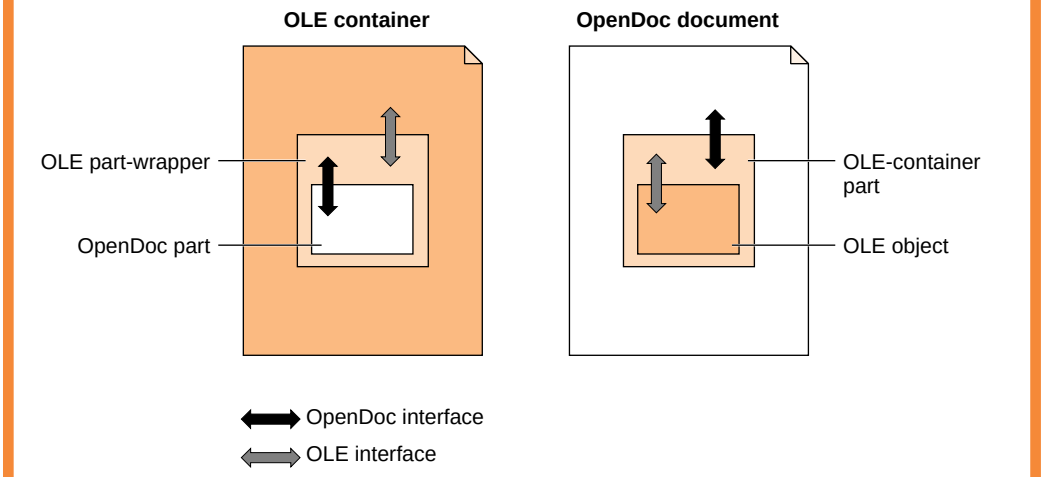
OLE 2.0 also relies on a different underlying object model than OpenDoc. Like SOM, the System Object Model (see the text box "The Object Underpinnings of OpenDoc" on page 18), the Component Object Model (COM) used by OLE supports dynamic linking and language-independent compatibility. Unlike SOM, however, COM does not support inheritance. Also, it is not CORBA-compliant, and thus OLE objects are directly compatible only with other COM objects.

The OpenDoc team, however, is currently building compatibility between OLE objects and OpenDoc parts on both Windows and Macintosh platforms, using a technology called Open Linking and Embedding of Objects (OLEO). With OLEO, users can embed OLE objects as OpenDoc parts in your OpenDoc documents and can likewise embed OpenDoc parts in OLE documents.

OLEO consists of two items: a portion of the OpenDoc document shell that functions as an OLE server, and a special OpenDoc part that functions as an OLE container. This is how it works (see also the figure "How OpenDoc-OLE interoperability works" on page 18):

- When a user embeds an OpenDoc part in an OLE container, the OpenDoc document shell functions as an OLE object (called an *OLE part-wrapper* because it's an OLE object that wraps an OpenDoc part) within the OLE container. The shell interacts with the part according to OpenDoc rules, and with the OLE container according to OLE rules.
- When a user embeds an OLE object in an OpenDoc document, the OpenDoc document shell automatically embeds the special part (called an *OLE-container part* because it's an OLE container that is also a part) around the OLE object. The OLE container part interacts with the OLE object according to OLE rules, and with its OpenDoc containing part according to OpenDoc rules.

OLEO is currently under development, and will be available on both Macintosh and Windows platforms shortly after the release of OpenDoc 1.0 on each platform.



How OpenDoc–OLE interoperability works.

and read the OpenDoc documentation to help get started. You will need to modify the .cpp file and include additional resources as necessary in the resources folder. You'll need to modify the project file if you change the file structure—for example, by breaking up the implementation into several files if your part editor is large.

You might, as a start, have your part display a different text string, or draw some other simple content element to the screen. Then, you might have it accept some sort of input to modify what it draws, perhaps by adding menu items or having it respond to keyboard input or mouse clicks. You might then give it more sophisticated capabilities, such as multiple facets, scrolling of its content, and additional kinds of content elements. You can give it storage capability, printing support, and so on.

The simple part you create with PartMaker cannot embed other parts, even though it can itself be embedded and it can also exist as the root part of its own document. Unless your parts are clocks or buttons or are otherwise restricted to one kind of content, you will eventually want to give them the capability of embedding other parts within themselves. Once you have done that, you will have created a part editor that is a full-fledged citizen of the OpenDoc world.

Create Container Applications With CALib

A container application is a conventional Macintosh application that has been modified so that it can embed parts and store their data in its own documents. If you already have an application, converting it to a container application is perhaps the fastest way to move into the OpenDoc world; it allows you to participate in the emerging OpenDoc market while you make your transition to OpenDoc parts. Once you have a container application, its documents can then handle any kinds of embedded parts, and the user will have full access to all OpenDoc features. About all the user will not be able to do is embed your documents as parts in other OpenDoc documents.

The Container Application Library (CALib), to be supplied with OpenDoc, is a code library that facilitates converting your conventional application to a container application. CALib gives you a procedural interface that completely shields you from the OpenDoc API. It makes your application look just like a containing part to parts that are embedded in your application. Also, it is designed to add as little as possible to the size of your application.

To use CALib, you make minor modifications to portions of your application's source code, including the addition of appropriate calls to CALib; then you recompile, and you're done. For example, your container application needs to be able to handle its own events, but it also needs to pass events to embedded parts so that they can draw themselves and

The Object Underpinnings of OpenDoc

It is impossible to know before run time what software will be needed to manipulate the parts of an OpenDoc document. Any system in which compound documents are constructed out of parts of indeterminate kinds must support dynamic linking of component software. In memory, OpenDoc parts are programming objects; the interfaces to those objects follow the specifications of the System Object Model (SOM),developed by IBM and used by OpenDoc on all its platforms. (See the text box on page 17, "OpenDoc-OLE Interoperability," for information on a competing object model.)

SOM not only supports dynamic linking, but is also programming language–independent. Users on a given platform can assemble documents out of OpenDoc parts whose part editors are written in any language and compiled with any compiler, without error. SOM objects follow the compatibility standards of the Object Management Group's Common Object Request Broker Architecture (CORBA), meaning that they can communicate with all other objects that are CORBA-compliant, even across platforms. SOM thus provides an interface standard for object interaction, that applies to binary (compiled and executable) code.

Perhaps the best advantage of SOM for you, the developer, is that SOM solves the "fragile base-class problem." Commonly, if the interface of any class in an object-oriented class hierarchy is modified in any way, all clients of the changed class (including classes derived from it) must be recompiled. For OpenDoc, that would mean that your part editors would function correctly only with the specific version of OpenDoc for which they were compiled. SOM corrects this by using a dispatching method that remains valid through changes to the interface or implementation of a class.

perform editing operations. CALib facilitates this by creating a proxy part to represent your application and by allowing it to interact with OpenDoc and embedded parts. You handle events in your normal fashion, but only after first passing them to OpenDoc so that embedded parts have a chance to receive them when appropriate.

CALib is under development and is not yet available. Preliminary versions of it are expected to be released before OpenDoc 1.0 ships. In the meantime, look for a folder in the OpenDoc for Macintosh Beta SDK containing more detailed information on CALib.

Use OPF for Fast (and Cross-Platform!) Development

OpenDoc supports the basic interactions required for parts to cooperate in constructing compound documents, but it is not itself equivalent to an application framework. It's not designed to shield you from the lower levels of system architecture, or construct complex resources or data structures for you. Nor does it relieve you from making platform-specific calls for some common tasks, such as drawing.

For truly fast and efficient development of part editors and high code reuse in cross-platform development, you should consider using the OpenDoc Parts Framework (OPF). OPF is a set of C++ class libraries that interact with the OpenDoc class library. You subclass and make calls to OPF classes in order to create your part editor. OPF automatically provides much of the default behavior that your part editor needs, and it also provides platform-independent graphics and memory-management models. Because of this, the total amount of code you have to write is far smaller than without OPF.

A big advantage of using OPF is that the default behavior it automatically provides is designed to be error-free and consistent with OpenDoc user-interface guidelines. But its greatest advantage is that you can develop for two platforms simultaneously, using the same set of source files. You can compile part-editor code built with OPF for either Macintosh or Windows platforms, using relatively small amounts of custom code for each platform. Almost all of the code you write is platform-independent.

OPF is under development and, like CALib, is not yet available. Preliminary versions of it have been seeded, and an alpha release should be available by the end of this year. Meanwhile, look for more information on OPF in a folder in the OpenDoc for Macintosh Beta SDK.

Start Today

Today, with OpenDoc at beta and OPF and CALib not yet available, your quickest way into OpenDoc programming is probably through PartMaker. It gets you up and running immediately, allowing you to create a functioning part editor in minutes. You can then concentrate your programming efforts on learning and applying the OpenDoc API, rather than struggling with development-environment rules and tracking down build problems.

OPF and CALib will be available soon, and for many kinds of development you will want to switch to them. Depending on how far you have taken your PartMaker-created part editor, however, you may also want to continue to develop directly to the OpenDoc API. PartMaker makes it easy to start programming, and the economical OpenDoc API makes it easy to add new features incrementally. ♣

Dave Bice (AppleLink address: BICE) is a member of of Apple's Developer Press and lead writer for the OpenDoc developer documentation.

PartMaker Output

When you run PartMaker and specify "MyPart" as the name of your part editor, you'll get a set of output files similar to these.

MyPart.μ	The project file (for developing with CodeWarrior).
MyPart.make	The MPW makefile (for developing with MPW).
ThePart.h	The C++ header file for MyPart.
ThePart.cpp	The C++ implementation for MyPart. Includes stubs or actual implementation for overrides of all methods of ODPart. For your convenience, this file includes debug strings and is extensively commented.
Config.h	A file you can use to make sure that only those parts of OpenDoc that you actually need are linked with your sample part.
•Resources	A folder that includes resource definitions for basic resources ('NMAP' resources, About-box dialog resources, the menu resource, plus all user-visible strings) needed by MyPart.
•Support	A folder of utility classes that you can use for tasks such as copying strings and managing lists of facets.
•SOM Support	A folder containing files that define a SOM class that wraps OpenDoc's ODPart class, allowing you to write your part editor's methods in a way that's more natural to a C++ programmer. You do not have to look at or modify anything in this folder.

CD Highlights

continued from page 13

containing the project's source code (for curious programmers who don't have the Newton Toolkit).

PCCTS

PCCTS (the Purdue Compiler-Construction Tool Set) is a public-domain tool set consisting of a parser generator called *ANTLR* and a lexical analyzer called *DLG*. ANTLR accepts a grammatical description for an input language and generates a recursive-descent parser in C or C++ to recognize sentences in that input language; DLG is used to break up the input stream of characters into a token stream.

ANTLR combines the flexibility of hand-coded parsing with the convenience of a parser generator. ANTLR has many features that make it easier to use than other language tools. Most important, ANTLR provides predicates, which let the programmer systematically direct the parse by means of arbitrary expressions using semantics and syntactic context; in practice, using predicates eliminates the need to hand-tweak the ANTLR output, even for difficult parsing problems. ANTLR also integrates the description of lexical and syntactic analysis, accepts LL(k) grammars for k>1 with extended BNF notation, and can automatically generate abstract syntax trees.

RAMDisk sample

This folder contains a sample RAM disk written in C. It can be compiled using MPW, Symantec, or Metrowerks compilers.

The sample 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.

ResEdit 2.1.3

ResEdit is Apple's standard direct-manipulation resource editing and creation tool. See the About ResEdit 2.1.3 file in this folder for a list of new features and bug fixes in this release.

ShrinkWrap™ 1.1

ShrinkWrap is a utility for creating and mounting DiskCopy images. It will also mount DART disk images and can utilize the StuffIt Engine for compression. See the file ShrinkWrap Read Me for more information.

Note: This is *not* an Apple product. It is provided on an "as is" basis. Apple is not responsible for any problems you may encounter in its use.

Snippets - AOCE

This folder contains new and revised code snippets, including the following:

- Construct AOCE Address: This snippet creates an AOCE packedDSSpec structure representing the address of the mailbox of the machine that it is running on. This could then, for example, be passed to SMPResolveToRecipient to create a mail address for the given machine.
- Create AOCE Business Card: This snippet shows how to create a PowerTalk business card anywhere on a given volume. Just create an FSSpec structure and call CreateBusinessCard() with it.
- DragBusinessCard: This program shows how to accept drag operations of flavorTypeDirectory and flavorTypeHFS for a PowerTalk business card, as well as drag operations from the application to the Finder to create generic clippings that contain an empty packedDSSpec structure.

Snippets - Networking

This folder includes the following new code snippet:

- ADSP Chat: ADSP Chat is a sample application demonstrating the use of the AppleTalk Data Stream Protocol (ADSP).

Snippets - Processes

This folder includes the following revised code snippet:

- LaunchWithDoc2: This is a document-launching sample program loosely based on C.K. Haun's LaunchWithDoc. This snippet includes some very useful routines.

Snippets - Sound

This folder includes the following new code snippet:

- MeterTest: This snippet demonstrates record metering through the use of SPBGetDeviceInfo() and SPBSetDeviceInfo() using the siLevelMeterOnOff selector.

Snippets - Toolbox

This folder includes the following new and revised code snippets:

- ColoredCheckBox: This snippet demonstrates how to create a checkbox on a gray window background by making sure that the background color for the window is set.
- GetDragHiliteColor: This snippet shows how to obtain the color that the Drag Manager uses to highlight regions when ShowDragHilite is called. Please note that this is only the current method for obtaining a highlight color; since it's undocumented, it will change in the future.
- GridWindowGrow: This snippet shows how to expand a window constrained to a grid—for example, to only allow a window to grow or shrink by 30 pixels.

SpriteWorld 1.0b4

SpriteWorld is a sprite-based animation architecture for the Macintosh. SpriteWorld consists of code libraries and interfaces that provide a simple but deep programming interface for implementing smooth, fast animation in your applications. In the interest of openness and learning, the full source code to the SpriteWorld libraries and sample applications is provided. SpriteWorld was designed for arcade game–style animation with full support for multiframe, overlapping, animated sprites and custom pixel-blitting routines. See the file SpriteWorld Read Me for details.

Next Month

Next month's CD will include practically everything in the known universe (including *develop* magazine articles) in subject-based DocViewer collections, new Developer Notes and Tech Notes, and more.

Alex Dosher
Developer CD Leader

Business & Marketing

Market Research Monthly

Surge in Small Office, Home Office Market To Continue

The small office, home office (or SOHO) market for personal computers in the United States is a market segment that Apple Computer, Inc., has been watching closely lately. We think it's a market you'll want to look at, as well.

Analysts look for certain conditions in markets to determine if they're ripe for generating new computer product sales. The first condition is a relatively low "penetration rate"—that is, the percentage of a particular market that already uses personal computers. Another necessary condition is a high need among customers in that market to use personal computers.

Both conditions exist in the U.S. SOHO market segment. (We define this segment as businesses with fewer than 20 employees, including self-employed, home-based workers.) "High need" is a given; perhaps more than others, workers in small business settings, particularly the self-employed, need the efficiencies and automation that personal computers can bring.

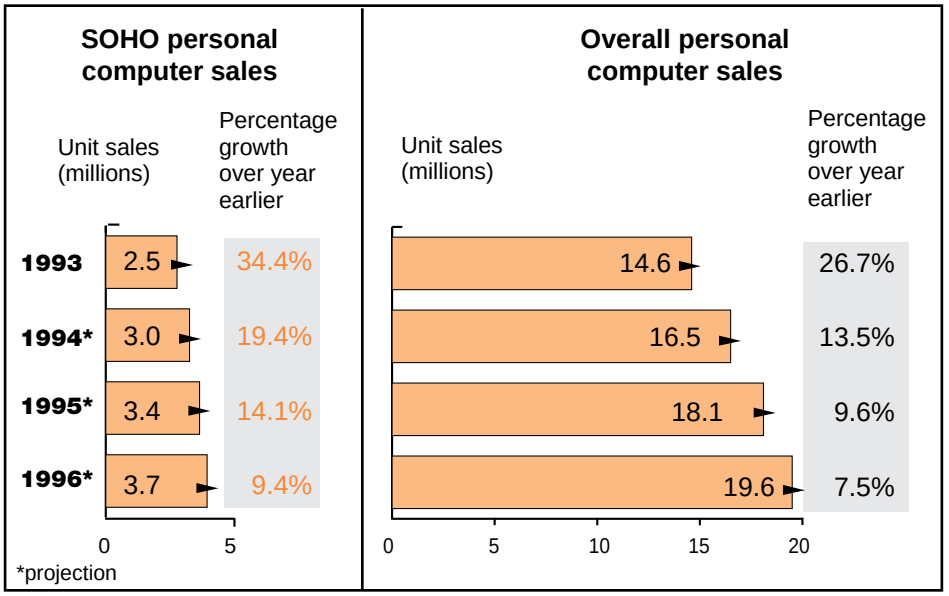
Further, relatively few home businesses—only about half of them—own a personal computers. By comparison, approximately three out of every four white-collar workers—fully 75 percent—in large U.S. business establishments use personal computers on the job; in certain white-collar intensive industries such as finance, insurance, law, and real estate, 80 percent of the work force uses a personal computer.

These two factors, undoubtedly along with recent computer price cuts, have led to a recent computer-buying surge in the SOHO market. In 1993, unit sales of personal computers in the United States increased 27 percent; sales to the SOHO market increased at a higher rate—34 percent according to Apple internal data. Apple expects this trend to continue at least through 1996. (See "U.S. Small/Home Office Personal Computer Sales, 1993–1996" on this page.)

The same phenomenon is occuring among Macintosh customers. In 1991 and 1992, the Macintosh SOHO market in the United States grew more than 60 percent faster than the total U.S. Macintosh market, according to Apple sales data. Apple's internal projections show U.S. SOHO

Inside This Section

SOHO Market to Outpace Overall Personal Computer Market



The small office/home office (SOHO) personal computer market in the United States will grow faster than the overall U.S. personal computer market between now and 1996, as shown by the "Percentage growth over year earlier" figures to the right of the bar graphs.

Macintosh sales continuing to outpace the overall market through 1998.

True, projections are just that—projections. But what Apple's forecast for the next several years shows is that, whether or not Macintosh sales to SOHO customers reach the targeted numbers, Apple will be spending significant marketing dollars to attract customers from small/home office settings to the Macintosh platform.

Reaching SOHO Customers With Your Products

The surge in the Macintosh small/home office market will provide many new buyers for your software and hardware products. You'll want to keep in mind the special needs of users in that market if you want to sell most effectively to them. To help you do this, we've digested data on the SOHO market from Apple's 1993 configuration study of the Macintosh installed base.

One might think that users in the SOHO market would be more likely to use higher end machines than average, because they're using their computers for business purposes. This assumption is mostly correct: SOHO customers are more likely to use modems, external hard drives, scanners, and removable media (such as cartridge drives) and to have more storage memory (80 megabytes) available than the average Macintosh user. On the other hand, on average they use the same amount of RAM as other Macintosh users (4 megabytes) and they're no more likely to have upgraded to System 7, use a CD-ROM drive, or be connected to a network.

In order of popularity, here are the kinds of applications Macintosh SOHO customers use and the percentage of those customers who use software in each category:

Word Processing	92%
DTP/Graphics	68%
Spreadsheet	67%
Accounting	65%
Database	64%
Education	43%
Drawing/Painting	48%
Games	42%
Presentations	38%
Calendar	35%
E-mail	19%
Prepress	17%
Programming	14%
CAD/CAM	12%
Multimedia	8%

We have to leave interpretation of this data to you, since we can't say what it means for your particular products or solutions. We can, however, urge you to look closely at the data, because there's no better time to sell into a market than when it's growing; all signs indicate that that's just what's happening to the small and home office market for Macintosh computers. ♣

Marketing Feature

A Copyright Primer for Multimedia Developers

By J. Dianne Brinson and Mark F. Radcliffe

New multimedia technologies make it easier than ever to combine film and television clips, music, graphics, photographs, and text into a software product. Unfortunately, these technologies *also* make it easier for you to infringe on someone's copyright without even realizing it. And ignorance of copyright laws could jeopardize your project's schedule and budget.

Multimedia law is a complex and evolving area of the law. As a multimedia developer, you often have to weave together bits and pieces of "content" plucked from radically different industries. Over the years, each of these industries has developed its own set of legal rules for intellectual property protection, leaving you and your lawyer to sort through stacks of legal tomes. In this article, we offer some ideas on avoiding copyright infringement in the United States and dispel some common copyright myths. This article is, of course, no substitute for professional legal advice. But it gives you, in plain English, enough information to speak intelligently to your lawyer or a content provider.

A Case Study

Perhaps the best way to illustrate the nuances of copyright law is to review the efforts of a fictitious multimedia developer, Productions, Inc., in creating an interactive training product called *You Can Do It.*

Let's assume that this product's script was written by a freelance writer. The product begins with an excerpt from a recording of Julie Andrews singing "Climb Every Mountain." It ends with a photograph of Lauren Bacall shown above the words "Good luck."

In this example, if the Productions staff didn't obtain permission to use the recording of "Climb Every Mountain" or the photo of Lauren Bacall, *You Can Do It* infringes three copyrights: the copyright on the song, the copyright on the Julie Andrews recording of the song, and the copyright on the photograph. Productions is also infringing Lauren Bacall's right of publicity (which is separate from copyright) by the commercial use of her image. Furthermore, if Productions didn't acquire ownership of the script from the freelance writer, Productions doesn't have clear title to You Can Do It, and its distribution may infringe the writer's copyright of the script.

So, what is Productions risking in this situation? If any of the copyright owners challenge Productions, the owners may be able to get a court order preventing further distribution of this multimedia product. And this type of revenue loss isn't one that most companies can afford to risk.

Three Steps to Avoiding Copyright Infringement

Here are three steps to help you avoid copyright infringement:

1. Assume that most of the third-party material you will want to use in your multimedia product is protected by copyright. At the beginning of your multimedia project, play it safe by budgeting time and resources for copyright issues. Most of the material that you use is probably protected for two reasons.

First, the types of third-party works that you'll most likely want to use—text, graphics, film and

Editor's note: This article is designed to provide information on copyrights. It is not legal advice. Don't make copyright decisions armed only with information contained in this article—obtain professional legal advice first.

television clips, music, and photographs—are the types of works that are protected by copyright. In copyright terminology, these are all "works of authorship." (Software and interactive multimedia works themselves are also copyrightable "works of authorship.")

Second, under current copyright law, it's easy to get copyright protection. Copyright protection arises automatically when an "original work of authorship" is "fixed in any tangible medium of expression." It's not necessary to file a copyright registration application to get copyright protection (although there are some good reasons for doing so). Most works easily meet the "originality" requirement, because a work is original in the copyright sense so long as it owes its origin to the author—that is, it was not copied from some preexisting work. A work can be original without being novel or unique. Only minimal creativity is required to meet the originality requirement, and no artistic merit or beauty is required.

A work can incorporate preexisting material and still be original. An example is a multimedia title that contains a clip from a famous classic film. When preexisting material—in this case, the film clip—is incorporated into a new work, the copyright on the new work covers only the original material contributed by the new author. And if the preexisting material is protected by copyright, the new author must get permission to use that material.

2. Analyze how you'll be using copyrighted material in your multimedia product. A copyright owner has five exclusive rights in the copyrighted work. These rights allow the owner to control use of the work by others:

• *Reproduction right.* The reproduction right is the right to copy, duplicate, transcribe, or imitate the work in fixed form.

• *Modification right.* The modification right (also known as the *derivative works right*) is the right to modify the work to create a new work. A new work that is based on a preexisting work is known as a *derivative work.*

• *Distribution right.* The distribution right is the right to distribute copies of the work to the public by sale, rental, lease, or lending.

• *Public performance right.* The public performance right is the right to recite, play, dance, act, or show the work at a public place or to transmit it to the public. In the case of a motion picture or other audiovisual work, showing the work's images in sequence is considered "performance."

• *Public display right.* The public display right is the right to show a copy of the work directly or by means of a film, slide, or television image at a public place or to transmit it to the public. In the case of a motion picture or other audiovisual work, showing the work's images out of sequence is considered "display."

With the widespread use of scanners and digital editing software, whole new copyright issues have arisen. The following example illustrates how what one developer might construe as "borrowing" a photographic image can result in several copyright infringements.

Example: A developer scans a photographer's copyrighted photograph, alters the image slightly by using digital editing software, and includes the altered version of the photograph in a multimedia encyclopedia that the developer sells to consumers. If the developer uses the photograph without permission, the developer infringes the photographer's copyright by violating the reproduction right (scanning the photograph), the

Five Common Licensing Myths

There are a number of myths out there concerning the necessity of obtaining a license for use of copyrighted materials. Here are five. Don't make the mistake of believing them:

Myth 1: "The work I want to use doesn't have a copyright notice on it, so it's not copyrighted, and I'm free to use it." Most published works contain a copyright notice. However, for works published on or after March 1, 1989, the use of a copyright notice is optional. The fact that a work doesn't have a copyright notice doesn't mean that the work is not protected by copyright.

Myth 2: "I don't need a license because I'm using only a small amount of the copyrighted work." It is true that *de minimis* copying (copying a small amount) is not copyright infringement. Unfortunately, it's rarely possible to tell where *de minimis* copying ends and copyright infringement begins. There are no "bright line" rules.

Copying a small amount of a copyrighted work is infringement if what is copied is a qualitatively substantial portion of the copied work. In one case, a magazine article that used 300 words from a 200,000-word autobiography written by former President Gerald Ford was found to infringe the copyright on the autobiography. Even though the copied material was only a small part of the autobiography, the copied portions were among the most powerful passages in the autobiography.

Copying any part of a copyrighted work is risky. If what you copy is truly a tiny and nonmemorable part of the work, you may get away with it (the work's owner may not be able to tell that your work incorporates an excerpt from the owner's work). However, you run the risk of having to defend your use in expensive litigation. If what you are copying is tiny, but recognizable as coming from the protected work, it is better to get a license. You cannot escape liability for infringement by showing how much of the protected work you did not take.

Myth 3: "Since I'm planning to give credit to all authors whose works I copy, I don't need to get licenses." If you give credit to a work's author, you are not a plagiarist (you are not pretending that you authored the copied work). However, attribution is not a defense to copyright infringement.

Myth 4: "My multimedia work will be a wonderful showcase for these copyright owners' work, so I'm sure they won't object to my use of their work." Don't assume this. Even if they're willing to let you use their works, they'll probably want to charge you a license fee. Content owners view multimedia as a new market for licensing their material. (The posting of copyrighted information on online bulletin boards has become a hot issue, so to be safe, get permission before you post another person's work.)

Myth 5: "I don't need a license because I'm going to alter the work I copy." Generally, you cannot escape liability for copyright infringement by altering or modifying the work you copy. If you copy and modify protected elements of a copyrighted work, you will be infringing the copyright owner's modification right as well as the copying right.

modification right (altering the photograph), and the distribution right (selling the altered photograph as part of the multimedia encyclopedia).

3. Get a license if you need one. You need permission—known as a *license*—to use a third party's copyrighted work if your intended use of the work would, without a license, infringe any of the copyright owner's exclusive rights. (See the text box "Five Common Licensing Myths" on page 23 for licensing issues of which you should be aware.) If you use material from someone else's copyrighted work in your multimedia product, at the very minimum you will have to copy the work. And if you copy the work without getting a license, you'll be infringing the copyright owner's exclusive reproduction right. Therefore, you need a license authorizing you to reproduce the material. You may need a license of other exclusive rights as well. For example, if you're licensing music for a video game that will be used in a video arcade, you'll need a public performance license. However, a video game used at home does not need such a license.

When You Don't Need a License—Fair Use, Public Domain, or Facts

You don't need a license in three circumstances: (1) if your use is "fair use"; (2) if the work you use is in the public domain; or (3) if the material you use is factual or an idea.

For the first circumstance, "fair use," it's often difficult to tell whether a particular use of a work is fair or unfair. The courts make such determinations on a case-by-case basis by considering four factors:

• *Factor #1: Purpose and character of use.* The courts are most likely to find fair use where the use is for noncommercial purposes. They are least likely to find fair use where the use is commercial. In fact, the Supreme Court has stated that commercial use is presumed not to be fair use. While it's possible to overcome the presumption that a commercial use is unfair, it's very hard to do so.

• *Factor #2: Nature of the copyrighted work.* The courts are most likely to find fair use where the copied work is a factual work or a work that has already been distributed. They are least likely to find fair use where the copied work is creative or fictitious, or the work has never before been published.

• *Factor #3: Amount and substantiality of portion used.* The courts are most likely to find fair use where what is used is a tiny amount of the protected work. They are least likely to find fair use where much of the protected work is used. If what is used is small in amount but substantial in terms of importance—the heart of the copied work—a finding of fair use is unlikely.

• *Factor #4: Effect on the potential market for or value of the protected work.* The courts are most likely to find fair use where the new work is not a substitute for the copyrighted work. They are least likely to find fair use where the new work is a complete substitute for the copyrighted work.

And, finally, if your multimedia work serves traditional "fair use" purposes—criticism, comment, news reporting, teaching, scholarship, or research—you have a better chance of falling within the bounds of fair use than you do if your work is sold to the public for entertainment and commercial purposes.

A Definition of Public Domain

You don't need a license to use a public domain work. Public domain works—works not protected by copyright—can be used by anyone. Because these works are not copyrighted, no one can claim the exclusive rights of copyright for such works.

The rules regarding what works are in the public domain vary from country to country. A work in the public domain in the United States may be protected by copyright in Canada or other countries.

There are several ways in which works fall into the public domain in the United States:

• *Expiration of the copyright.* A copyright that was in existence before January 1, 1978, and was renewed, has a term of 75 years. All copyright terms run to the end of the calendar year in which they expire. Consequently, in 1994, all works first "published" before January 1, 1919, are in the public domain in the United States. In some countries in Europe, documents enter the public domain 50 years after the death of the copyright owner.

• *Failure of the copyright owner to renew the copyright.* Under the 1909 Copyright Act, copyright protection lasted 28 years. A copyright owner could obtain an additional term, known as a renewal term, by filing an application to renew in the 28th year. The Copyright Renewal Amendment of 1992 eliminated the requirement of filing a renewal application for works published between 1964 and 1977, inclusive. Renewal is not required for works created after 1977. However, before 1992, a number of works entered the public domain because the copyright owner failed to file a renewal application.

• *Failure to use a copyright notice on publicly distributed copies of a work (for works published before March 1, 1989).* Under prior law, the distribution of copies without copyright notices resulted in the forfeiture of copyright protection. For works distributed before January 1, 1978, forfeiture was automatic. For works publicly distributed after that date, the copyright law provided ways around the defect created by distribution without notice.

No License Required—Ideas, Facts, or Your Own Works

You don't need a license to copy facts or ideas from a protected work. The copyright on a work does not extend to the work's facts. This is because copyright protection is limited to original works of authorship, and no one can claim originality or authorship for facts.

Naturally, you don't need a copyright license for material that you create yourself. However, you should be aware that the rules regarding ownership of copyright are complex. You shouldn't assume that you own the copyright if you pay an independent contractor to create the work (or part of it). In fact, generally the copyright in a work is owned by the individual who creates the work, except for full-time employees working within the scope of their employment and copyrights that are assigned in writing.

The issue of copyright ownership has been raised in a lawsuit between two leading multimedia developers, Michael Saenz and Joe Sparks. They are currently in court in a dispute over whether Sparks was an employee or an independent contractor of Reactor, Inc. (Saenz's company) when Sparks helped to write their famous game, "Spaceship Warlock." If Sparks is right in claiming that he was an independent contractor, he is co-owner of the copyright and has a right to half of the proceeds from the game. The lesson to be learned? Make sure copyright issues are clear before you hire contractors.

Copyrights—Take a Conservative Approach

If you're going to be successful in the multimedia industry, you need to understand the basics of copyrights and several other laws that govern this area. Ignorance of copyright laws can be costly. At the very least, having to get copyright licenses late in your project cycle could delay your ship date and impact your budget. And, worst case, if you illegally use copyrighted material, you could be forced to stop product shipments and could incur liability for millions of dollars in damages.

The best strategy is to take a conservative approach. Early in your project, budget time and resources to copyright issues. And to be safe, obtain professional legal advice before you ship your product. ♣

This article is based on material from the Multimedia Law Handbook *by J. Dianne Brinson and Mark F. Radcliffe, a partner at Gray, Cary, Ware, and Freidenrich. This book is available from Ladera Press in Menlo Park, California, (800) 523-3721. Copyright 1994 by J. Dianne Brinson and Mark Radcliffe.*

Developer Outlook

A Tale of Three CDs

Nontraditional Developers Use the Apple Media Kit to Enter New Markets Rapidly

By Kris Newby

It's been a little over a year since the first version of the Apple Media Kit shipped, and the fruits of this multimedia integration tool are just beginning to line the retail shelves. One of the signs that this tool is truly easy to use is the number of developers outside the traditional software industry who have been able to create sophisticated database-driven CD-ROM titles.

In this article, three nontraditional developers—an architectural firm, a publisher of women's reference titles, and a dating service—talk about how the Apple Media Kit helped them create their first CD-ROM titles in under a year. Though these companies serve very different markets, they were all drawn to this tool for similar reasons:

• *Easy cross-platform development.* The Apple Media Kit enables developers to quickly create products on the Macintosh, then compile the results for playback on Macintosh and Windows-based computers. By placing Macintosh and Windows versions of a title on a single CD, developers can use the same media elements for the programs, making the most of a CD's available storage. And they can sell to two markets with only one SKU (shelf-keeping unit).

• *A compatible creative process.* More multimedia developers are finding that the creative process used in industries such as filmmaking and music recording works well for developing multimedia titles. The Apple Media Kit facilitates this process, enabling editors, media artists, and programmers to work in parallel. This approach saves development time and alleviates the need for artists to learn scripting or programming languages.

• *A fast-turn authoring environment.* Compared to other authoring tools, the Apple Media Tool—the authoring part of the kit—is very easy to learn and use. A drag-and-drop metaphor enables users to quickly design screens and add interactivity.

• *A powerful object-oriented programming language.* The Apple Media Programming Language (the other part of the Apple Media Kit) enables developers to extend and enhance a program beyond what can be done with the Apple Media Tool alone. In the case of three developers discussed in this article, the object-oriented nature of the programming language will allow them to reuse much of their database and interface code in future titles.

Read on to hear more about what inspired these developers to create their first CD titles and about the marketing and technical factors that influenced their decision to use the Apple Media Kit for title development.

1994 Builder Magazine Electronic Buyer's Guide

Terry Beaubois, RDC Interactive Media, Palo Alto, California

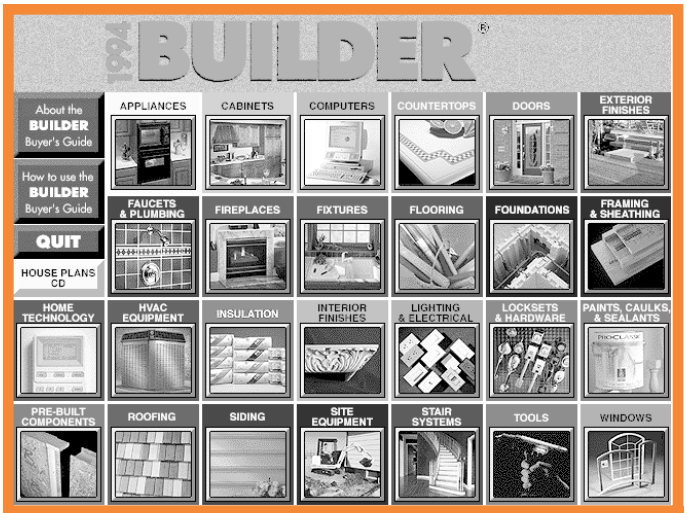
Terry Beaubois was working as a residential design architect when the idea for a new CD-based product came to him.

"My firm used Macintosh computers extensively in our residential design process, but we could only computerize our business up to a certain point," said Beaubois. "We hit a wall when it came time to select specific building components. To find the dimensions of a specific sink or window, we'd have to dig through stacks of dog-eared catalogs or call manufacturers' showrooms. I realized that what architects really needed was all this information on one convenient CD."

Twelve months after finding a publishing partner for his project, Beaubois and his computer-savvy architect partner, Greg Miller, shipped the solution to this problem—*The 1994 Builder Magazine Electronic Buyer's Guide.* This CD provides architects, builders, and do-it-yourselfers with a database of over 1,750 manufacturers and 5,200 building products, from plumbing fixtures to windows. Using the vendor lists compiled for their annual *Builder Magazine* manufacturers issue, publisher Handley-Wood provides manufacturers with free address and telephone listings on the CD—or, for an additional fee, lets them include ad screens, product catalogs, distributor maps, and even videos of new products.

Selling the Idea

To find a publishing partner for this CD idea, Beaubois and Miller developed a rough prototype, then began knocking on the doors of building industry manufacturers. Their first tries weren't successful. Manufacturers liked the concept, but no one wanted to risk being the



Using the Apple Media Kit's cross-platform capabilities, RDC quickly created a combined Macintosh/Windows CD for *Builder Magazine.*

first to go "electronic." Finally, their search for a publishing partner led them to *Builder Magazine.* The CD provided advantages to both parties: It allowed the publisher to reuse the data from their annual manufacturers issue, and it saved Beaubois and Miller the legwork required to get product information from over a thousand companies.

"We realized that using a magazine metaphor for our CD would help building industry acceptance," said Beaubois. "The industry had an established protocol for magazine advertising that could be moved over to this new medium, providing a low-risk way for the publisher to make money."

Avoiding the "C++ Cellar"

Once they found a publishing partner, Beaubois and Miller began evaluating development tools. Their objective was to find an easy-to-use Macintosh-based authoring tool that had facilities for designing a cross-platform product, since 80 percent of *Builder*'s readers are Windows users. These requirements narrowed their options considerably. SuperCard and HyperCard were crossed off the list, since they aren't cross-platform tools. And at that point Macromedia was still just talking about adding cross-platform capabilities to their Director product. So they took a chance on the newcomer to the field—the Apple Media Kit.

Greg Miller summarized his favorite aspects of the Media Kit: "It's easy to create cross-platform products, it allowed us to quickly integrate media elements together, and it enabled me to keep track of our more than 2,000 screens with no difficulty."

Because of the complexity and memory footprint of their project, they eventually hit the technical limits of the authoring part of the Media Kit, and they had to do some custom programming in the object-oriented Apple Media Programming Language. They used the language to optimize performance and run-time size. They also added features such as an object-oriented database engine, scrollable "hot spots" that linked mouse clicks to the database, a Windows installer, and an active "go to main menu" key on every screen.

Cross-Platform Issues

Beaubois and Miller also appreciated the ease with which they could create a hybrid Macintosh/Windows CD, on which all the media elements are shared. But as avid Macintosh users, they made a startling discovery: There's a tremendous amount of confusion surrounding "multimedia-ready" computers on the DOS and Windows side of the business. In fact, it took them two weeks to find a Windows user who could play a CD.

Beaubois mused, "We figured we spent over a month doing Bill Gates's job—finding out what a multimedia player really is in the Windows world, and then informing our Windows-based customers what kind of system is needed to play our CD. The reality we had to face was that when Windows users put our CD in their machine and it doesn't work, they think it's our fault—even though the problem may stem from the fact that their system doesn't have a sound card or speakers."

Future Plans

Looking ahead to next year's version of the *Builder Buyer's Guide,* Miller will again use the Apple Media Kit. "Last week we did a project using Director just to try it out, but we still prefer the Media Kit because it works best with our creative process. We use a 'Hollywood model': We assemble the best possible team of freelancers; they all work concurrently using their favorite rendering or illustration tool; then they hand off finished media to our integration person."

When asked how this CD will affect his future business plans, Beaubois replied, "I'm going to continue doing a mix of multimedia and architecture. It's not so strange if you think about it. Multimedia actually requires an aptitude in the same three areas as architecture: art, business, and science. In the long run, I think our CD is making a greater contribution to the field of architecture than just designing another house. One of the things I'm looking forward to is the possibility of doing a project that uses the Media Kit and Apple's upcoming QuickTime VR (virtual reality) software. As an architect, this technology greatly enhances my ability to communicate spatial relationships and information about buildings. I'd love to use VR to design a product that really transforms the way architects work."

Her Heritage: A Biographical Encyclopedia of Famous American Women

Chip Canty, Pilgrim New Media, Cambridge, Massachusetts

Pilgrim New Media is creating a line of CD-based material that focuses on an underserved segment of the software market—women. Pilgrim's first product, *Her Heritage: A Biographical Encyclopedia of Famous American Women,* lets users explore the lives of more than 1,000 famous women through text, photos, and film clips.

"We saw that there was a strong need for more titles that interest women," said Chip Canty, chairman and CEO of Pilgrim New Media. "And since women don't tend to use computers for entertainment, we wanted to deliver an entertaining, informative product that didn't look like a game."

Before Canty began his first title, he assembled a strong, well-rounded team that included a marketing person, a programmer, a graphic designer, a filmmaker, and Robert McHenry, editor-in-chief of *Encyclopaedia Britannica*. Pilgrim then licensed a library of women's biographical information from Merriam-Webster and formed an editorial advisory board that included women's studies

included scholars, educational television producers, and librarians, to name a few.

Canty, who previously worked in marketing at Interleaf, knows how to sell traditional software products, but his new product requires a different approach. Since women are more likely to spend time in bookstores than computer stores, Pilgrim chose a distributor, Cambrix Publishing, that works with bookstores to encourage them to carry CD-ROM titles.

Fast Cross-Platform Development Required

From August to December 1993, Pilgrim's Chief Technical Officer George Touchstone conducted a search for the best development tool for their CD titles.

"We immediately realized we needed almost 3,000 screens to display all our biographies," said Touchstone. "We had enough experience with established multimedia authoring tools to know that a product with so many screens would take too long to code and revise. To save development time we looked for a tool with which we could create a single 'template' interface for displaying our information from a custom database engine."

For marketing reasons, Pilgrim decided it had to ship both a Macintosh and Windows version of the product on one CD. And cofounder Michael Sullivan, a graphic designer, felt strongly that they needed to use a tool that could create visually appealing screens. The ease with which all these goals could be met using the Apple Media Kit was the most important factor in their choice of this tool.

"The Apple Media Kit enabled all the text files, pictures, and database information to be shared by the Macintosh and Windows run times," said Touchstone. "We kept our run times small so that we could fill the rest of the CD with the good stuff."

A Database-Driven Title in Eight Months

Canty's team used the Media Kit Programming Language to extend the product beyond what the authoring tool alone could offer them. They designed their database engine and interface so that with a few cosmetic changes, they could reuse it for other titles. The object-oriented nature of the kit's programming language makes this possible: All of a screen's objects are created programmatically, and then text, graphics, and movies are substituted in real time to this screen from an underlying custom database of media objects.

The programming language also enabled Canty's team to create a new text object class that gave them more flexibility in styling the text within their Macintosh and Windows versions. And for the Macintosh environment, they designed the product so that it checks a system's memory and, if there isn't enough, it automatically reduces the viewing resolution and raster depth.

"We only began prototyping in December 1993. In five weeks we had a working prototype, and we were able to announce our product in June and ship by September," said Canty. "If we'd followed the usual model, we would've had to freeze editorial work early and hand the project off to engineering and production. With the software we built using the Apple Media Kit, our editorial staff, media integrator, programmers, illustrator, and designer could work in parallel, coming together every couple of weeks to update the code and rebuild the database in a batch process. And the last-minute updates—as when Jessica Tandy, the actress, died just days before we shipped—could be handled as smoothly as the earliest edits, without rewriting any code."

Pilgrim will be able to repurpose the custom database and interface used in their *Famous American Women* CD for future titles.

Pilgrim's Promise—More Women's Studies Titles

Pilgrim currently has three more projects in the works, and they're planning on making their publishing engine technology available on a contract basis to other publishers, corporations, and institutions that need an efficient means of converting printed materials into an interactive multimedia format.

And, of course, Pilgrim will continue to produce more women's history titles modeled after *Her Heritage.* They anticipate that these future titles will be significantly quicker and easier to produce using the database engine they created with the Apple Media Kit—because, rather than having to recode the whole project, they'll only have to change the text in the underlying database and the screen template.

CD-ROMANCE: The Multimedia Meeting Place

Howard Brummer, Romulus Productions, Carmel, Indiana

When Howard Brummer left his position as chief information officer at a national real estate development company, he called Peter Weisz, a longtime friend who worked in the multimedia business, to review potential multimedia business ideas. After sifting through numerous concepts, they settled on a CD-based product idea that was near and dear to Brummer's heart—a dating database.

Brummer, who recently reentered the "dating scene," could see the need for an alternative to personal ads and singles bars. The solution: His company's soon-to-be released title, *CD-ROMANCE.* With this CD, singles can evaluate potential dates based on photos, video clips, and information about their interests. CD users can expedite the process of searching through more than 300 potential dates in the database by specifying criteria such as age,

It took Romulus nine months to take their dating CD from concept to shipping product.

religion, location, and height in the database's search engine.

Finding a Tool Match

Romulus's first choice for a development platform was the Macintosh, but to maximize their revenues, they felt they needed to sell to Windows users too. After researching development options, they chose the Apple Media Kit as the best cross-platform development tool and selected The Carl Group in Cupertino, California, to design their product from the ground up. The founder of this programming service bureau, Tim Carl, summarizes the reasons his company used the Apple Media Kit for this project:

"If you want to do something interesting in multimedia, the best approach is to author it on the Macintosh platform, then port it to Windows. The Apple Media Kit enables us to provide our customers with a faster cross-platform development cycle than other tools, and by using its object-oriented programming language, we can add unique features like CD-ROMANCE's date-selection interface."

Brummer also likes the fast-turn development cycle that the Apple Media Kit makes possible. "From a marketing standpoint, we liked the fact that, using the Apple Media Kit, we can ship both our Macintosh and Windows versions on a single CD, and that changes to our database can be made without disturbing our code and our schedule."

It took Romulus about nine months to go from a concept to a shipping product. They produced a prototype in five weeks and a working prototype in three months, then spent the remainder of the time fine-tuning their product and gathering media from advertisers.

Romancing the Media

In July Romulus sent out evaluation copies of *CD-ROMANCE* to the press and testers, and since then they've been deluged with orders and media exposure, including coverage on MTV news and other television news shows.

"I think that we were lucky to come up with a product idea that piqued the media's interest," said Brummer. "In addition, we hired a media consultant to help us with the launch. And we employed various marketing tactics such as sending e-mail product announcements to the Internet addresses of key media contacts and video news releases to television networks."

But as a new developer selling a single title, getting the interest of large distributors hasn't been easy. As a result, Romulus will sell their product through mail order, direct-mail ads, and smaller distributors. This should change over time, however, since Romulus's future plans include more CDs, including geographically focused versions of *CD-ROMANCE*.♣

Kris Newby, principal of Kris Newby Technical Communications, is a marketing communications consultant and freelance writer based in Palo Alto, California.

Ordering the Apple Media Kit

The Apple Media Kit consists of the Apple Media Tool, an authoring application, and the Apple Media Tool Programming Environment. The Apple Media Tool is available for $795; the Media Tool Programming Environment costs $995; and the kit, which includes both the tool and the programming environment, costs $1,495. You can currently obtain the Media Kit through APDA by calling (800) 282-2732, and in the near future you'll be able to get it through other retail channels. Technical support is available through Apple Computer, Inc., by calling (800) 767-2775. Qualified higher education customers should contact their local education representative for information about special educational-use pricing.

To make licensing simpler and more affordable, Apple has bundled the run-time licensing of the Apple Media Kit with QuickTime and QuickTime for Windows. This single license, valid for three years, is available for enterprise and commercial publishers at a flat fee of $500 per title. For educational institutions and nonprofit organizations, there is no royalty fee.

APDA Ordering Information To place an APDA order from within the United States, contact APDA at (800) 282-2732; in Canada, call (800) 637-0029. For those who need to call the United States APDA office from abroad, the number is (716) 871-6555. You can also reach us by AppleLink; the address is APDA.