The Developer Business Report *November 1995*

AppleDirections

Inside This Issue

Editor's Note: Microtrends	2
IndustryWatch: The Macintosh Strategy	3
Support for OpenDoc Builds With International Endorsements	10
New 100-MHz Power Macintosh for Education	10
QuickTime 2.1 Offers New Features	11
Apple and Digitool Ship Macintosh Common Lisp 3.0	12
Apple Resumes Shipping PowerBook 5300 Series, Reduces Prices	12
Apple, H-P Announce Macintosh Application Environment for Japanese Market	13
Copland Developer Release Update	13
CD Highlights: Tool Chest Edition, November 1995	14
OpenDoc Human Interface FAQs	15
Optimizing PowerPC Programs	17
Special Marketing Report: Developer Opportunities in the Japan Market	23
Confessions of a New Macintosh Developer	27
Developer University Schedule	30
The Internet Page	30

Apple News

Apple Redoubles Developer Support Efforts

Includes Reduced Pricing, More Internet Support

As Acting VP of Apple Developer Relations Shirley Stas put it recently, "What's *not* new this month?"

Late last month, at the Software Publishers Association (SPA) conference, Apple announced a sweeping set of changes that greatly improve your access to development tools and support. These include

- expansion of the Apple Associates Plus program to selected markets outside the United States
- reductions of up to 60 percent in key Apple-labeled development tools and services when purchased from APDA locally or in the United States
- automatic inclusion of all the technology SDKs (software development kits) currently on the Mac OS SDK CD to all program members
- posting of technology SDKs on Apple's FTP site, making them available free to anyone who wants them
- expanded services and development information on the Internet
- automatic seeding of new Macintosh software technologies to Macintosh program members

please turn to page 9

Strategy Mosaic

Apple's Development Tools Strategy

Apple and Others Deliver on the Promise

By Paul Dreyfus, Apple Directions *staff*

The shipment in September of Oracle Power Objects under an Apple label may not mean much to you, unless you're a client/server database developer. However, as the French say, *"Au contraire, mon frere."*

The point I'll be making in this month's column is that Apple's release of Oracle Power Objects for both Mac OS and Windows systems—coupled with a variety of other recent tools announcements from Apple and other vendors—*is* significant to you. Together, they make the Macintosh computer a far more robust development platform. For the past year or so, Apple Computer, Inc., has undertaken a concerted effort to turn the Macintosh computer into a solid development platform, one that you'll want to use regardless of whether you develop for Windows or Mac OS systems. It's easy to throw around terms like *industry-best* and *state-of-the-art;* suffice it to say that the recent tools releases have made it much easier to be enthusiastic about developing on—and for—the Macintosh computer.

I'll describe some of these specific tools a little later, and I expect you'll find them inter-

please turn to page 4

AppleDirections

Volume 3, 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 (SZABLA)

Production Editor
Lisa Ferdinandsen (LISAFERD)

Contributors
Paul Baldwin, Alex Dosher, Peter Green, Gary Moulton, Bob Megantz, Kris Newby, Kerry Ortega, Geoff Schuller

Manager, Developer Press
Dennis Matthews

Manager, Apple Developer Periodicals
Mark Bloomquist

Production Manager
Diane Wilcox

Prepress/Film
Aptos Post

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

Editor's Note

Microtrends

What do rock-and-roll music, beer, and the Internet have in common? To me, recent trends in those three areas represent victories of individualism over conformity, of creativity over corporate marketing, of substance over style—all things folks at Apple and in the Apple developer community have traditionally stood for.

Take rock-and-roll music. Have you checked out CD stores and music magazines lately? There are more bands performing and making CDs than you can shake a drum stick at. There have always been lots of bands, but now it seems that many more of them are "making it," at least to the level of recording and selling their music. There's also an incredible variety of commercially available music—rock, hip-hop, blues, heavy metal, grunge, rap, rhythm and blues, and so on.

It feels like a big difference from not so many years ago, when it seemed as if large "corporate" rock-and-roll, embodied by groups like the Rolling Stones, the Eagles, and Pink Floyd, dominated popular music. Certainly corporate rock rolls on, but now if you walk into a big CD store, you just won't believe the choices that are available, some of them even recorded by independent labels.

If you like what's happening in rock, you'll love what's going on with beer, at least in the United States. It seemed only a few months ago that U.S. supermarkets and liquor stores stocked about three different kinds of beer, namely the ones advertised during TV sports. Now you can find literally dozens of different kinds of beers made by microbreweries from all over the United States, not to mention a far greater selection of beers from around the world. These are beers with *character,* definitely not your typical pallid American brew. Interestingly, the amount of beer that's drunk in the United States is staying about the same, which means that these new beer makers are actually cutting into the sales of the huge U.S. breweries.

These new business directions in music and beer are quite commercial—no one's doing this for nothing. But for the microbands and microbreweries suddenly marketing their wares, it seems like the effort comes from the heart—the desire to make interesting music and beer—rather than from a heartless corporate marketing department.

I think the same is true of the Internet, especially the World Wide Web. It may be the one area of our business that defies large corporate control and centralization. The world's communications conglomerates can merge, strategize, and figure out just how to control the airwaves. In the meantime, real people can put information on the 'net in whatever form they want, and they can find what they're looking for in the way that best suits their styles and needs. (I've heard it said that Microsoft views its greatest competition as the Internet; good luck to the folks in Redmond if they want to try to control that as well.)

Maybe someone will come up with an electronic equivalent of the Dewey Decimal system for online posting and retrieval (although I hope not). Or maybe one of you out there is just clever—and fiendish—enough to control the Internet. But it's my bet that all efforts to unify, standardize, conglomerate, or in some other way control the 'net will last just as long as it takes a hacker to break through a new security system.

What this means is that there will remain huge opportunities for little guys to provide real people with the tools they want for posting information, retrieving it, organizing it, and communicating it to others. Apple is certainly seizing on this opportunity with its Web server solution, Cyberdog (a group of OpenDoc parts that let users build their own way of using the Internet), and other products. I think you'll want to seriously explore ideas for Internet products of your own. By giving your customers what they really need to access the Internet, you probably won't come to dominate the industry. But you'll be doing your part to assure that the 'net remains one of the bastions of creativity and individual expression—and potentially making a good business by doing so.

Paul Dreyfus
Editor

IndustryWatch: News & Perspective

From the Macintosh Platform Marketing Group: The Macintosh Strategy

[Editor's note: We mentioned last month the series "Macintosh vs. Windows 95" that Apple has been preparing to help slow, if not derail, the Windows 95 juggernaut. Michael Mace and the Macintosh Platform Marketing group have just published the twentieth and last in this series, entitled "The Macintosh Strategy," which we're reprinting here, pretty much as is. There's no question that this is Apple's view of the personal computing market. We're featuring it so you can think about how you fit into that vision, and how you can help bring it about. To see previous entries in the series, check out the location http://www.apple.com/whymac/ on the Internet's World Wide Web.]

The first 19 updates in the series "Macintosh vs. Windows 95" explain various Macintosh advantages over an Intel-based personal computer with Windows 95. There are still a lot of Macintosh advantages that we didn't cover in this series—we haven't even touched on Macintosh versus PC video capture, for example. And we didn't say anything about the relative cost of phone support. (If you buy a Macintosh in the United States and many other countries, you get a free 800 technical support number that you can call as much as you want, for as long as you own your computer. To get help from Microsoft, you either call a 900 number that costs $1.95 per minute or you call a long-distance number and pay toll charges. You might ask yourself why Apple can afford to give away free technical support and Microsoft can't—maybe Macintosh users don't need as much help as Windows users? Or maybe Microsoft is just running short on cash.) We'll report from time to time about this and other issues.

This note steps back and describes how all those advantages add up to fundamentally change personal computing. Anything Apple says about its direction tends to suddenly show up in the feature plans of certain other computing companies, so we won't be too specific about what's coming. But in general terms, here's where Apple thinks the marketplace is going, and where it's looking to take the Macintosh platform.

December *Apple Directions* Online

The December issue of *Apple Directions* will be available on AppleLink, the Internet, and eWorld by November 15, at the following locations:

AppleLink: path—Developer Support:Developer Services:Periodicals:Apple Directions.

Internet: select Developer Services and Products at the location www.apple.com.

eWorld: in the Apple area of the Computer Center.

Technology: The Transition to Next-Generation Computing

The industry is entering two critical technology transitions that will eventually render previous standards obsolete. On the hardware side, everyone in the industry is moving beyond old-style CISC chip technology, on which Pentium, for example, is based, to new architectures that include new instruction sets. The only disagreement in the industry is over what to call the transition. Apple and most other companies call it RISC. Intel calls it "VLIW" or "Beyond RISC." The name isn't critical. The important thing is that we all agree the industry is moving toward new instruction sets, forcing recompilation and rewriting of existing programs.

On the software side, the industry is just beginning the transition to component software (also called *object-oriented software*). Component software is important because it breaks large operating systems and programs into small modules that are much more easily updated, debugged and changed around. It's the key to the next generation of computer software because it will allow programs and operating systems to be customized more easily, and will enable smaller developers to get back into software development.

Going along with the two technology transitions is a major evolution in the way computer customers think. The general-purpose computer market is dying. Just as the automobile market moved from general-purpose transportation to market segments in the 1920s, the computer market is moving into a much more segmented world today. For example, the needs and desires of home users are very distinct from those of business, and the differences are growing rapidly. One of the biggest challenges for computer companies in the 1990s is delivering very different products to different people.

Apple's Strategy: Drive the Transitions

The combination of new hardware and component software is essential to deliver the next generation of features that computer users want. A high-performance architecture such as RISC is essential to really make multimedia, 3D, speech recognition, and other leading-edge technologies become mainstream. And the flexibility of component software is required if we're to customize computers for different markets.

Apple believes that the Macintosh platform is for delivering the next generation, because Apple is the only computer company that can change the hardware and software in tandem. So Apple has already made the transition to next-generation hardware. On the Intel PC side, the hardware transition is just starting with the move to Intel's P6 chip, which reportedly runs many existing applications slower than a Pentium. And beyond that, there's a lot of uncertainty over what other changes P7 and other chips may bring.

In component software, Apple is investing heavily with a number of major partners—including IBM, Novell, Oracle, Adobe, and others—to create an open, cross-platform component software product called

OpenDoc. Shipment of OpenDoc on the Macintosh computer is scheduled for the very near future.

And in terms of the market, Apple is now focused on creating specific products and solutions that address the needs of different types of customers. Some people have erroneously called this a move into market niches. Apple doesn't view it that way—in reality, the center is evaporating, and in the fairly near future only market segments will be left. Apple's move into segmentation gets the Mac platform ahead of that trend.

Making the Transition Easy

The last time Apple made a transition to a new generation of computers (the 1984 move from the Apple II to the Macintosh), Apple made a number of errors that it's committed not to repeat. First, Apple failed to give its installed base of loyal Apple II users an easy growth path into the next generation. With the move to Power Macintosh, Apple has given its new computers the ability to run old-style Macintosh software, so investments are protected and customers can move easily.

Second, in 1984 Apple failed to make it easy for DOS users to move to its new systems. This time, Apple's correcting that by providing a number of products that make it easy to use DOS and Windows files, and even run DOS and Windows programs, on a Power Macintosh. With this change, DOS and Windows users have, for the first time, a realistic alternative to the Intel/Microsoft standard.

Third, in 1984 Apple failed to license other companies to make Mac-compatibles. Without alternative sources, prices remained too high, and some customers were uncomfortable making a commitment to the Mac platform. Apple has decisively changed that situation by licensing Mac-compatibles, some of which are shipping today. With the 1996 advent of the Common Hardware Reference Platform (the shared base hardware design codeveloped with IBM), very open licensing will be in place.

How It All Adds Up

One of the most common errors people make when evaluating the Macintosh strategy is to look at one part as if it's the whole thing. So they'll say that Apple's strategy to make the Macintosh platform successful is licensing. Or that the strategy is PowerPC. In reality, the strategy is the sum of all those initiatives together: The combination of leading the technology transitions (with PowerPC and OpenDoc), driving more segment-focused marketing, changing the Macintosh business model to embrace compatibility and openness, and of course innovating so the Macintosh computer becomes even more attractive to users. The synergy between those initiatives is much more powerful than any one of them would be alone.

What About the Future?

It's common in the computer industry today to look on Windows 95 as if it's the endpoint in the evolution of personal computing. Computers have now been perfected; everyone else can just pack up and go home. In reality, the interesting changes are just beginning, and Windows 95 is at best a rest stop on the road to the next generation of personal computing (apparently it doesn't even take full advantage of Intel's latest chip). No doubt eventually the PC companies will sort out this and many other transitional issues they face in the next few years. Someday you'll be able to get next-generation computers from the PC crowd. But Apple believes it can deliver the benefits of the next generation much sooner, with a smoother transition, and with more attractive features. And this time Apple is not keeping the new generation to itself.

Questions or comments? You can send e-mail to the Macintosh Platform Marketing team at competition@applelink.apple.com. ♣

Strategy Mosaic

Apple Strategy

continued from page 1

esting, because they can save you time and money and potentially make your programmers more creative. But first, let's look at the bigger picture of Apple's overall development tools strategy. (Much of what follows is based on a long conversation I had with Mike Zivkovic, manager of Apple tools business development and one of Apple's tools strategists.)

"Normal" Tools Development

A brief analogy will help me make my point. In his influential work on the history and philosophy of science, Thomas Kuhn identified two types of scientific discovery: evolutionary science and revolutionary science. Most often, scientific research proceeds along *evolutionary lines:* One small finding begets another, and the scientific community proceeds with one hypothesis, one experiment, and one finding at a time toward some greater understanding of nature.

Once in a great while, though, a great thinker like Isaac Newton or Albert Einstein (or Guy Kawasaki?) comes along with a way of viewing the gathered body of scientific data that's so different and so compelling that it causes *revolutionary* change in the scientific community. In a very short period of time, a revolution changes the way all scientists approach their work. Afterwards, scientists go on with their "normal" work, within the new paradigm established by the revolutionary thinkers.

Apple is fond of announcing the revolutions in computing that it's trying to bring about. You'll probably be glad to hear that I'm not writing to announce yet another revolution. In fact, at least in the area of tools, Apple is undertaking normal, evolutionary, one-tool-at-a-time work to fulfill the same strategy—with a couple of minor alterations—that it formulated last year. (And, by the way: When's the last time you can remember Apple having the same tools strategy within a 24-month period?)

This work isn't dramatic, and it's not going to make anyone wake up one morning with a new way of looking at the world. But it is making it much easier and more attractive to develop software on the Macintosh computer and for the Mac OS platform.

The Apple Tools Strategy: Same as It Ever Was

When Gregg Williams last wrote about Apple's tools strategy in the August 1994 installment of Strategy Mosaic, he described the following three main goals:

- to give you a wide choice of traditional development tools at a reasonable price
- to develop and make available new development tool technologies that will improve your productivity by a quantum leap
- to provide cross-platform development tools that allow you to reach the Mac OS and Windows platforms (for a complete summary of the Apple tools strategy of a year ago, see page 1 of the August 1994 issue of *Apple Directions*)

In my conversation with Mike Zivkovic, it quickly became apparent that the basic strategy has remained the same, with some changes in emphasis. Apple's development tools strategy, as of September 1995, has the following three main goals, as articulated by Mike:

- *strategic goal #1:* to provide you easy access to Macintosh technology, so you can make it available to our mutual customers through the products you create
- *strategic goal #2:* to offer a complete range of cross-platform tools that enable deployment of software not only to the Mac OS and Windows platforms, but also to Pippin, Newton, and interactive television systems
- *strategic goal #3:* to make programmers more productive

I think you'll agree that these goals are more similar to the goals of a year ago than they are different. By working toward the same overall strategic goals, Apple's brain trust has enabled tools developers inside and outside Apple to carry out "normal" development, filling your development tools needs at a rate that's lately built up a head of steam.

Did you know that in the late summer and fall of this year alone there have been nearly 30 Apple tools-related announcements? Some of these, such as OpenDoc Developer Release 3, a pre-release version of OpenDoc Development Framework, Newton Toolkit 1.5, and the Apple Media Tool CX Media Database Extension, have come from Apple. But the bulk of them, and many of the tools that I think will spark your interest, have come from other vendors. (For a list of recent announcements, see "Tools-Related Announcements, Summer–Fall 1995" on page 6.)

Different Tools for Different Developers

Apple's tools strategists realized some time ago that Apple couldn't provide all the tools you need, let alone the ones you want. As part of the foundation underneath the strategy, Apple is working with third-party vendors to assure that the Mac OS platform remains the platform of choice for existing Mac OS developers and becomes the platform of choice for new ones.

In both its own tools development and its work with other tools vendors, Apple focuses on delivering tools in these four main areas:

- application development tools, which need to be powerful and productive and give programmers the ability to manage increasingly complex programs
- client/server and database development tools, which need to let their users deliver applications that can be scaled from the least powerful PowerBook computers to the most powerful desktop systems and from small workgroups to larger networks; these tools also need to enable computers from different environments (Mac OS, Windows, OS/2, UNIX®, and so on) to be able to work together in the same network
- solution integration tools, which need to allow rapid, scalable deployment of solutions that can be easily customized to meet specific needs
- multimedia authoring tools, which enable easy integration of sound, video, animation, and 2D and 3D graphics

Progress Toward Strategic Goals

Apple's primary goal when it comes to tools—strategic goal #1 mentioned earlier—is to provide you with an easy way to give your customers access to Macintosh technologies. Note that some of these Macintosh technologies, such as QuickDraw GX, QuickDraw 3D, and QuickTime VR, may seem like development tools in and of themselves. In the Apple view of the world, these are *technologies,* not tools: In order for customers to access their features, they must be present on customers' systems. Strictly defined, development tools let you give your customers access to the various Macintosh technologies, but the tools themselves don't have to be present on the users' systems for your products—and the Macintosh technologies—to work.

To fulfill that goal, Apple and third-party vendors provide a great many tools, giving you choice about the tools you use,

Cross-Platform Tools

Primary development platform	Tool type	Products
Macintosh, deploying to Windows	Cross-compilers	Code Warrior
	Cross-platform frameworks	ODF, Zinc, OM++
	Multiplatform authoring	Apple Media Tool, Oracle Media Objects, Director
	Multiplatform client/server	Oracle Power Objects, Forte, JYACC JAM, Omnis 73 PowerBuilder, Uniface, Unify
Windows, deploying to Macintosh	Porting tools	Altura, MS Visual C++ Cross Dev (680x0 only), ICE OM++
Macintosh/Windows simultaneously	Multiplatform compilers	Symantec C++, Prograph, MicroFocus COBOL, True Basic, Visual Works, SmalltalkAgents
	Multiplatform client/server	Oracle Power Objects, Forte, Omnis 73, PowerBuilder, Uniface, Unify, JYACC JAM
	Multiplatform GUI builders	XVT Development Solution, Neuron Data Open Interface, Galaxy, Zinc
Macintosh, deploying to Newton	Application and content tools	Newton Toolkit, Newton Book Maker
	Forms tools	OmniForm/Wright Strategies, Sestra PowerForms, TabulaRasa/BrandX Software

no matter what kind of developer you are. The latest *Guide to Macintosh Development Tools,* which you can find on this month's edition of the Developer CD, lists more than 200 specific tools.

Fulfilling strategic goal #2 (giving you cross-platform development capabilities), nearly 50 of these tools offer multiplatform deployment. The latest development in this area is Metrowerks's latest release of Code Warrior, which provides a cross-compiler that you can use to easily compile either Mac OS or Windows software. For a list of these tools, including the type of development they're used for and the platforms they help you to target, see the chart "Cross-Platform Tools" on page 5.

The Significance of Oracle Power Objects

In carrying out strategic goal #3—making programmers more productive—Apple and its development tools partners are making things really interesting. For example, Apple's MrC compiler produces high-quality code that's optimized for PowerPC microprocessors, saving your programmers the time and trouble of optimizing by hand to get maximum performance. What's more, in the near future you'll be able to use it directly within several of the most often used C/C++ Macintosh development environments provided by Apple and other vendors.

Another new wrinkle to Apple's tools strategy is to borrow key components from MPW and make them available in other environments. MrC is the first product of this genre; we'll let you know about others as Apple is closer to shipping them.

It's in the area of productivity that the release of Oracle Power Objects (OPO) is genuinely significant for client/server and database developers and symbolically significant for the rest of you. OPO is a visual programming tool, which means that instead of having to use a complex programming language, you select from a broad menu of visual items to create software. OPO is useful only if you're creating relational database client/server solutions. But it's pretty darned useful: Instead of having to write in the SQL programming language, you simply drag and drop onscreen

Tools-Related Announcements, Summer–Fall 1995

Here is a partial list of development tools shipped in the late summer and early fall by Apple and third-party vendors. It includes information and development tools for Apple platforms.

Apple Announcements

- OpenDoc Developer Release 3, which includes the following:
  - –OpenDoc Frameworks pre-release version
  - –Novell Appware
  - –CALib updated version
- MPW Pro/E.T.O. #18, which includes final candidate versions of the following:
  - –MrC
  - –MrC++
  - –Power Macintosh debugger
  - –MPW Command Reference in QuickView
- Mac OS SDK #4 includes the following new software development kits (SDKs) and software:
  - –QuickTime Conferencing SDK
  - –QuickDraw 3D SDK
  - –QuickTake camera software
- Apple Media Tool CX, with CX Media Database Extension
- Newton Toolkit 1.5
- Newton Book Maker 1.1
- New NewtonScript compiler and profiler
- Updated developer support programs, including pay-as-you-go code-level support from Developer Technical Support and substantial price reductions for many Apple tools and support products (see "Apple Redoubles Developer Support Efforts" on page 1 of this issue for details)
- New Apple Multimedia Program
- Free on-line access to all Apple developer-related information through the World Wide Web at http://www.apple.com

Third-Party Announcements

- Oracle Power Objects (Apple-labeled product ships)
- Metrowerks Code Warrior Gold 7, with Macintosh/Windows cross-compiler as well as tools for 680x0 processor–based and PowerPC processor–based Macintosh, Magic Cap, and Windows systems
- New Symantec C/C++ plug-in compilers; final candidate software can be found on MPW Pro/E.T.O. #18
- Pictorius Prograph CPX, new native PowerPC version that delivers both 680x0 and PowerPC executables
- Pictorius Peregrine, built on top of Prograph CPX, a visual environment for building client/server database applications
- Quasar Knowledge Systems SmalltalkAgents Professional version 2.0
- Digitool Macintosh Common Lisp 3.0 (see "Apple and Digitool Ship Macintosh Common Lisp 3.0" on page 12 of this issue)
- JYACC previews JAM 7 cross-platform client/server application tool for Macintosh computers

Many of these tools are available from APDA, Apple's source of development tools. See page 32 for APDA ordering information.

objects—representing database fields and tables, for example—to build your application. What's more, applications developed using OPO are immediately deployable on Mac OS and Windows systems, and they support Oracle, Sybase, and Microsoft SQL servers.

The next version of OPO, scheduled for release in early 1996, will be even more useful. It will be written in "native" PowerPC code, making it faster and more productive than the current version, and it will include native-mode support for Power Macintosh versions of OPO-based software. It will also include OpenDoc container support, meaning that any OPO-based application can function as an OpenDoc container; you'll be able to add features to an OPO-based application by dragging OpenDoc parts into it.

Further, it will support OS/2 as well as Microsoft's Open Database Connectivity (ODBC) protocol for network middleware; this last feature means that it will support virtually every current database management system. You can read more about OPO—especially how it's an improvement over Microsoft's visual programming tool, Visual Basic—in the OPO white paper that's posted at the *Apple Directions* Web site (go to www.apple.com and select Developer Services and Products) and in the September 1995 issue of *Apple Directions,* also available at the Web site.

Productivity Through Visual and Dynamic Environments

What's significant about OPO is the fact that it's a visual programming environment, one of a growing number of Macintosh tools that can let you and your programmers make more efficient use of your time and money. It's been said that the future of personal computer programming is in visual and object-oriented, dynamic environments—environments that don't require the use of complex languages and/or enable different parts of the development process (design, coding, testing, debugging, and so on) to take place simultaneously, rather than in sequence. OPO and a few other tools suggest that we're making great progress toward that future.

One of those tools is Pictorius's Prograph CPX, a visual programming, native PowerPC

Dynamism and Inertia

By Dave Johnson

Recently I took a class in Newton programming. The language that you use to program the Newton, NewtonScript, is an example of an object-oriented dynamic language, or OODL. I don't pretend to be an expert in languages, not by a long shot, so I can't offer any incisive comparisons with other "modern" languages, but I *can* tell you what it feels like for a dyed-in-the-wool C programmer to leap into this new and different world. It feels *great.*

One well-known feature of dynamic languages is garbage collection, the automatic management of memory. Objects in memory that are no longer needed are automatically freed, and in fact there is no way to explicitly free them other than making sure that there are no references to them any more, so that the garbage collector can do its thing. I didn't fully realize how much time and effort and code it takes to deal with memory management until I didn't have to do it anymore. There's something almost naughty about it, going around cavalierly creating objects in memory without worrying about what to do with them later. After a lifetime of living in mortal fear of memory leaks, it feels deliciously irresponsible. I like it. I like it a lot.

With dynamic languages like NewtonScript, you can let go of the details of the machine's operation, and deal with your program's operation instead—you can think at the design level, not the machine level. And it's an incredible relief to float free of the bits and bytes and pointers and handles and memory leaks and messy bookkeeping. Most of the ponderous baggage that comes along with writing a computer program goes away. I mean really, how much longer must we approach the machine on its terms when we want to build something on it? Users were released from that kind of bondage to the machine's way of doing things long ago. So what are we waiting for?

Obviously we can't program the Macintosh in NewtonScript (more's the pity), but why aren't we all chucking our C++ compilers in exchange for Prograph or Lisp or Smalltalk or Dylan? Well, some of us are. But I think there are two major hurdles to overcome before dynamic languages become mainstream: the need for speed, and inertia.

Dynamic languages carry their own baggage, of course. In the same way that making the Macintosh easier for people to use made it harder to program because the complexity and bookkeeping were shunted behind the scenes, making programming languages easier to use also requires new behind-the-scenes infrastructure and complexity. (*Somebody* has to do the memory management, after all.) This usually results in a bigger memory footprint and slower execution.

Inertia is the other big problem. People, once they know one way to do something, are often loath to change it, especially if they've been doing it that way for a long time. I'm guilty of this in my own small way: every time I learn a spiffy, liberating new way to program I think I'll never go back to the "old" way. But the next time I set off to write some code I automatically reach for the familiar tools, not the new ones.

Fortunately, neither one of these hurdles will stop the evolution of our tools. It's unstoppable, if perhaps slower than we might like. There's already a whole spectrum of tools available. From Assembler to AppleScript, Pascal to Prograph, there are tools that allow anyone with enough interest to teach their computers to do new things. The line between users and programmers continues to blur, and dynamic languages can only help that process. I love the thought of putting programming tools into the hands of "nonprogrammers"—kids, artists, hobbyists—and seeing what they come up with. You can bet it will be something new, something that people tied to the machine would never have thought of. I can't wait.

This article is excerpted from "The Right Tools for the Job" Dave Johnson's Veteran Neophyte column for the forthcoming Issue 24 of develop, the Apple Technical Journal. develop *Issue 24 will be posted on eWorld, AppleLink, and the World Wide Web by December at the same locations where you can find* Apple Directions. *(The locations are listed on page 3 of this issue.)*

environment for use by professional Mac OS application developers. It provides all the power of C++ in a completely visual environment that simply doesn't let you make syntactical programming errors. You can still make design and logic errors, but you don't have to know even the least bit of C to use it (although you can import existing C code into Prograph-based applications). Currently, Prograph CPX creates 680x0-based and PowerPC processor–based executables; it doesn't yet support cross-platform development, although a new version is expected to change that, and provide OpenDoc support as well. Developers who have worked with it report vast productivity increases over traditional C programming environments.

Another development environment with the potential to make programmers more productive is Quasar Knowledge Systems' SmalltalkAgents Professional, a fully developed object-oriented Smalltalk environment. Since it's fully dynamic, you can edit code while you're running it without having to recompile, saving a great deal of time over traditional programming methods. SmalltalkAgents Professional includes tools for designing, developing, and managing frameworks, classes, and objects, and also provides tools for building user interfaces and other parts of applications by using Macintosh Drag and Drop.

Advanced Tools = More Programmers

In describing these specific products, I'm neither giving them the *Apple Directions* seal of approval, nor recommending that you run right out and buy them. I mention them because their very existence, not to mention the creativity of the people who invented them, suggests just how vital a platform the Macintosh computer is, and how important it could become to the future of personal computing. Such tools include not only OPO, Prograph, and SmalltalkAgents Professional, but also the following, as well:

- Novell's AppWare, a completely visual tool for custom business solution software
- Dylan, Apple's dynamic programming language, which is expected to be available in a special technology release later this year
- the forthcoming Denali, a visual OpenDoc parts builder that Apple will ship early next year

These advanced tools can potentially broaden the number of programmers by a significant factor. The number of hot-shot, specialized C programmers is actually decreasing in relation to today's growing personal computing industry. Engineering schools are turning out the same number of genius programmers each year, yet each year software gets more complicated and there's greater demand for innovative solutions.

Visual and object-oriented, dynamic environments let people with less-specialized training get into programming, thereby increasing the ranks of those who can develop computer software. These environments also let specialized programmers work more quickly. Dynamic languages enable programmers to throw out the traditional edit-compile-link cycle associated with static languages and work more quickly. Visual programmers are likewise freed of the somewhat onerous task of having to write all their applications' features in C or some other text-based language.

The proliferation of OPO, Prograph, and other like environments will have the following results:

- There'll be a larger talent pool of potentially less expensive programmers, and, depending on who you are and the kind of development you do, you'll be able to spend less on programming and more on design, usability testing, marketing, and other aspects of your business.
- You'll be able to get products into the marketplace more quickly than before.
- You'll be able to continue to produce increasingly complex software without having to spend so much on it that no one can afford it.
- The software industry collectively can become more innovative, with more people able to contribute their ideas more easily and quickly than before.

Normal Developments, Extraordinary Results

These results are not a sure thing. Mike Zivkovic, who's a great fan of visual and dynamic programming, admits that there are reasons these tools haven't caught on as much as they might have. Talking about Prograph CPX, he says, "It's a little unusual to see the visual aspect of a program rather than the code while you're programming."

In fact, visual programming is unusual enough that many, if not most, of today's programmers remain more comfortable working with a text-based language, despite the obvious performance boosts of visual programming. Similarly, programmers who grew up on a diet of static languages are just more comfortable with the edit-compile-link cycle, even if it is more burdensome than the efficiency offered by a dynamic language. (For an interesting point of view on this subject—and on another dynamic environment, NewtonScript—see the box "Dynamism and Inertia" on page 7.)

I'm not trying to predict the future in this article. Instead, I'm saying that by deliberately sticking to its tools strategy for over a year, and by putting one foot in front of the other and almost intentionally not trying to make great leaps forward, Apple and its tools allies have delivered. The tools that are available for the Macintosh computer make it a hugely viable, economical, and productive development platform, one that shows great promise for letting you deliver on the promise of Macintosh technology.

For Complete Macintosh Development Tools Information

For more information on Macintosh development tools, you'll want to read the Fall 1995 edition of the *Guide to Macintosh Development Tools* on this month's Developer CD. You can obtain additional information from the following sources:

- on eWorld—The Developer Corner:Apple Developer Services:Tools Demos & Technical Overviews
- on the Internet—http://www.info.apple.com/dev or ftp.info.apple.com/Apple.Support.Area/Developer_Services
- from the Developer Support Center at 408-974-4897 or Internet address devsupport@applelink.apple.com
- in the Apple *Developer Tools Catalog* (call 716-871-6555; or use Internet address apda@applelink.apple.com, America Online address APDAorder, or CompuServe address 76666,2405).

Remember my brief analogy in the beginning of this article? In formulating his ideas about the history of science, Thomas Kuhn paid particular attention to the transition from "normal" evolutionary science to revolutionary science. What enables these great thinkers and their great ideas to change science, seemingly overnight, are the weight of many individual discoveries, made during the "normal" phase. These smaller findings provide data on which the new scientific paradigm rests; they also begin to get people used to the new way of looking at the world and the universe.

We may be witnessing such a transition in the area of programming tools. Apple, Oracle, Pictorius, Novell, and too many others to list by name here, are doing the one-day-at-a-time grunt work of delivering on the Apple tools strategy. Perhaps their "normal" development work, especially in the area of visual and dynamic environments, is laying the foundation for a true revolution in programming—one in which the vast majority of programming is done in these new, more productive, and potentially more creative environments. ♣

Apple News

Developer Support

continued from page 1

- a new Developer University mini-course (on writing device drivers) on the World Wide Web
- regular marketing information from Apple in your e-mail box through the new Apple Directions Express list server

The Apple Associates Plus Program

Historically, only Apple Partners have been able to get technical help (through e-mail) from Apple Developer Technical Support (DTS) engineers. This situation improved in the United States and Canada earlier this year when Apple created a new program called the Associates Plus program. This program allowed developers to get limited DTS support (up to ten technical, API-level, or code-level technical questions answered per year) and to receive the Mac OS Software Development Kit.

All Apple developer program members will soon receive the contents of the Mac OS SDK as part of their memberships (see "Wider Distribution of SDKs," later in this article), but the expanded availability of the Apple Associates Plus program is good news for you if you live in Germany, Austria, the United Kingdom, France, Finland, Norway, Sweden, Denmark, Estonia, or Iceland. If your direct-support needs are modest, the Apple Associates Plus program can save you money by giving you direct DTS support for considerably less money than Apple charges for the Apple Partners program.

Seeding to Non-Partner Members

Until recently, only Apple Partners who had signed NDAs (nondisclosure agreements) regularly received beta software from Apple. Apple is now extending this privilege to Apple Associates and Apple Associates Plus members in most of Europe. Upon receipt of a signed NDA, Apple Associates and Apple Associates Plus members will receive the same Macintosh Technology Seeding CDs that are regularly sent to Apple Partners.

Price Reductions on Tools and Services

Apple has made significant price reductions worldwide in key tools and services offered through the Apple *Developer Tools Catalog* (see the box on page 8). This should make it easier for all developers, especially those on a limited budget, to get the tools they need to develop on Apple platforms. You can order the tools listed in the table through your local APDA location at a significant discount, or you can order from APDA in the United States and receive the prices quoted in the table on this page.

Wider Distribution of SDKs

The Mac OS SDK CD includes SDKs for over 30 system extensions—including such key technologies as Apple Guide, AppleScript, Macintosh Drag and Drop,

Product	Old price	New price
QuickTime VR 1.5 and MPW Bundle (NTSC or PAL)	$695	$595
MPW Pro	$495	$295
MPW Pro Updates	$195	$75
MPW Pro set of manuals	$299	$149
E.T.O. complete new subscribers package	$1,095	$795
Annual E.T.O. subscription renewal	$400	$250
Mac OS SDK CD (and renewal)	$299	$149
Apple Developer Mailing (and renewal)	$250	$149
Apple Directions subscription	$99	$49
VITAL Technical Architecture Guides	$300	$200
AppleScript Software Developer Toolkit 1.1	$199	$99
Virtual User 2.0.1	$150	$79
Apple Media Tool 1.2 (U.S. and Int'l)	$495	$475
Apple Multimedia Info Mailing (and renewal)	$400	$300
Newton Developer Mailing (and renewal)	$200	$149

APDA offers key products at lower prices. The prices listed here are for the United States market only.

the Thread Manager, and QuickTime.

Starting in December, when the next Mac OS SDK CD is due, all the SDKs from that CD will be sent to all developers worldwide as part of the Apple Developer Mailing. (Currently, Apple Associates must buy it separately from Apple for $149.)

In addition, Apple will be placing the SDKs for as many technologies as possible on the Apple FTP site. This will make these technologies available, worldwide, to anyone who has Internet access.

More On the Internet

Apple is constantly working to improve the quality, quantity, and usefulness of developer-related information on the Internet—all of which is available, free of charge, to anyone with Internet access. Here are some of the things Apple is doing on the Internet:

- Apple continually adds files available for downloading from the Apple FTP site. Apple has converted almost all of its documentation to Adobe Acrobat format, which means you can now use one application to view all electronic documentation from Apple.
- The http://www.info.apple.com Web site includes links to pages on key Apple technologies such as OpenDoc, QuickDraw 3D, and PowerTalk. These pages include everything from basic "what is it?" information to technical documentation, marketing materials, and even downloadable software.
- Apple has considerably improved the Technical Q&A's home page, which now includes better formatting of the question-and-answer pairs, as well as the ability to do boolean and wildcard searches on the Q&A database. Another place to go for quick access to technical subjects is the Tech Info Database web page, located at http://www.info.apple.com/til.html.
- Apple's Web site also contains the full content of both the current and back issues of *develop* and *Apple Directions* magazines.
- Apple maintains a searchable directory of third-party products on the Web; you can use this directory to get free publicity for your products. To add your products to this database, go to location http://www.info.apple.com/dev/thirdparty/third_party.html.
- Apple Developer University now has five introductory courses on the World Wide Web: What Is OpenDoc?, Introduction to PowerTalk, Introduction to RISC Technology, Programmer's Introduction to PowerPC, and Writing and Using Device Drivers (the newest). If you have access to the Internet and a World Wide Web browser, you can learn about key Apple technologies without leaving your office.

Apple Directions Express

Apple has started a listserver that will bring a newsletter of high-level news and information to your desktop on a regular basis. This listserver is the fastest way that Apple has to get useful information to you. It's short, so it won't be a burden to read, but it will have pointers to in-depth information on Apple's World Wide Web pages and other sources. To subscribe, send e-mail to adirections@thing1.info.apple.com; in the body of the message, type *subscribe <your real name>*.

For More Information

The changes to developer support listed in this article have come about largely because of your input, and future improvements will be based on your feedback. For more information, contact your local developer-support program manager.

You can ask questions or get further information in several ways. First, you can call the Developer Support Hotline in the United States at 408-974-4897, or you can write the Hotline at DEVSUPPORT (AppleLink) or devsupport@applelink.apple.com (Internet).

Second, you can read more about various developer support services by going to Apple's home page at http://www.info.apple.com and clicking the Developer Services button.

Support for OpenDoc Builds With International Endorsements

Companies worldwide—not the least of which are Japanese—are expressing support for the OpenDoc view of component software.

Component Integration Laboratories, Inc. (CI Labs), the vendor-neutral industry association promoting component software technology, recently announced that Oracle Corporation has joined as a full member. Last summer, Oracle announced that its first OpenDoc implementation of Oracle Power Objects will ship this fall. (See "Apple to Offer Oracle Power Objects for Client/Server Development" on page 12 of the September 1995 issue of *Apple Directions*.)

Another recent piece of news is the exchange of memberships between CI Labs and the IntelligentPad Consortium (IPC). IPC is an international consortium founded in Japan that promotes the IntelligentPad technology, a component-based media architecture. It is a vendor-neutral, nonprofit, privately funded organization with over 50 members including companies, institutes, and prominent individuals from Japan and worldwide. According to John Cheuck, director of IPC global planning and promotions, "Technology is creating a global marketplace, making it imperative that software architectures interoperate. The membership exchange between the IPC and CI Labs is a step towards making this a reality."

Last summer, CI Labs announced its commitment to work with the PC Open Architecture Developers' Group (OADG). OADG, a consortium of more than 220 Japanese hardware and software manufacturers, will cooperate with CI Labs to introduce and promote the OpenDoc component software architecture in the Japanese PC industry. OADG includes such companies as Acer Japan Corp., ALPS Electric Co., Digital Equipment Corporation Japan, Intel Japan K.K., Hitachi Ltd., Fujitsu Ltd., Justsystems Co., Mitsubishi Electric Co., Sony Co., and Toshiba Co.

Also last summer, CI Labs announced 25 new members, six of whom are based outside the United States, including companies in Canada, England, Germany, and the Netherlands.

New 100-MHz Power Macintosh for Education

In its continued pursuit of Power Macintosh market share in the education market, Apple Computer, Inc., recently announced the availability of the Power Macintosh 5300/100 LC, a PowerPC processor–based multimedia workstation for education. This announcement follows the April 1995 introduction of the Power Macintosh 5200/75 LC, which was

the first Power Macintosh designed specifically for education.

The Power Macintosh 5300/100 LC features a 100-MHz PowerPC RISC 603e processor with 256K Level 2 Cache, 16 MB of RAM (expandable to 64 MB), and a 1.2 GB hard disk drive. It also includes an Apple SuperDrive floppy disk drive, an internal AppleCD 600i Plus tray-loading quadruple-speed CD-ROM drive, and two expansion slots—a Macintosh LC processor-direct slot and a communications slot (for an internal modem or Ethernet card). Built-in multimedia technologies include a video output connector, a video capture card, a TV tuner, and 16-bit stereo sound input and output capabilities.

The Power Macintosh 5300/100 LC is available immediately. The system is priced at U.S. $2399 for direct sales to education, and is also available with setup for U.S. $2459.

For further information, customers in the United States should call the Apple Education Hotline at 800-800-APPL. Customers outside the United States should contact their local Apple office for information. You can find more information on Apple Education's Web site at location http://www.info.apple.com/education.

QuickTime 2.1 Offers Important New Features

Despite its ".1" suffix, QuickTime 2.1 for Macintosh offers significant improvements over QuickTime 2.0, including the ability to play "sprite graphics" and better audio and video playback. QuickTime 2.1 is the most recent component added to the QuickTime architecture, the only cross-platform multimedia architecture that allows developers to work with many multimedia data types—including MPEG and other video formats, various sound formats, MIDI music sequences, virtual reality scenes, still images, motion JPEG, and CD-quality sound—and work with them all as QuickTime movies. This allows developers to focus on creative content instead of on the mechanics of integrating different technologies.

This article describes various noteworthy features of QuickTime 2.1 and tells you how to obtain the QuickTime 2.1 software.

Software Sprites

QuickTime 2.1 for Macintosh adds sprite tracks to QuickTime movies. A movie using a sprite track can be more highly compressed than a traditional QuickTime animation, since each sprite image is only included once and the sprite track itself contains only the sprite's path. Because the sprite animation is contained in a movie, sprite movement is automatically synchronized with the other movie tracks (most commonly, sound and video).

Modifier Tracks

QuickTime 2.1 for Macintosh includes a new modifier track feature, which allows you to make your movies more dynamic. For example, instead of playing video in the normal way, a video track can send its image data to a sprite track. When the movie is played, the video track appears as a moving sprite.

You can use a modifier track to store a series of sound volume levels, thus allowing you to dynamically adjust the volume of a sound track. Similarly, you can use modifier tracks to store location and size information, enabling a video track to move and resize as it plays.

Enhanced Sound on Power Macintosh Computers

Sound Manager 3.1, included with QuickTime 2.1 for Macintosh, provides significantly enhanced recording and playback performance on all Power Macintosh computers. Because Sound Manager 3.1 can be up to several times faster than the previous version in handling sound, a Power Macintosh computer has more time for playing QuickTime video tracks; this results in higher frame rates and smoother overall video playback and recording.

Audio Improvements

Sound Manager 3.1 also provides two new audio compression formats for 16-bit sound: IMA and μLaw. The IMA 4:1 audio compression format is based on an Interactive Multimedia Association standard. The μLaw 2:1 format is an international standard for compressing voice-quality audio (typically 16-bit, 8-kHz speech), often used in telephony applications and on the Internet as the encoding format for "AU" sound files.

Using QuickTime 2.1 for Macintosh, Sound Manager 3.1, and any QuickTime-aware application (such as Apple's MoviePlayer or SimpleText), a consumer can now transparently open and play ".WAV" and ".AU" format sound files, commonly found on the World Wide Web.

Sound Manager 3.1 also provides a plug-in architecture for audio compressors and decompressors. This will enable third-party audio-compression vendors to integrate their products into the QuickTime architecture.

Video Improvements

Using a new technique for authoring video content, QuickTime 2.1 for Macintosh enables content developers to create video that is larger and looks better on most computers, yet still plays back well on slower systems. Now consumers can play movies at full-screen size on most Mac OS computers. On entry-level computers, the video can be played back at half-size (which is larger than the quarter-screen size previously recommended).

Previously, images compressed with Cinepak, the most common digital video compression format in use today, have been stored in a format using millions of colors. However, most computers only have 256-color display devices, resulting in noticeable degradation of image quality when Cinepak movies are played. QuickTime 2.1 for Macintosh adds a 256-color format to Cinepak, enabling high-quality video playback without any color remapping.

CD-ROM AutoStart

It's not a new feature (it's been present since QuickTime 2.0), but we thought we'd mention it again. QuickTime includes a feature called AutoStart, which allows you to create CD-ROM discs that automatically start up when they're inserted. In particular, you can configure your CD so that it automatically opens any application or document that is at the root level of a Macintosh HFS-formatted CD.

Other Improvements

- *User control of text.* QuickTime 2.1 for Macintosh adds a settings dialog box so users can control the font, font size, style, and color of the text. Text tracks, part of QuickTime since version 1.5, are frequently used as captions for video, song lyrics, and more.
- *Full-screen support.* QuickTime 2.1 for Macintosh provides a convenient way for developers to easily change the monitor

resolution within their title so the content fills the entire screen.

• *QuickTime Conferencing support.* QuickTime 2.1 for Macintosh, with its support for compressing audio as it is captured, is used by QuickTime Conferencing—Apple's videoconferencing solution—to reduce the amount of data required to transmit high-quality audio.

• *Apple MPEG support.* QuickTime 2.1 for Macintosh supports the Apple MPEG Media System, providing expanded MPEG support for multimedia developers.

• *Support for professional digital video cards.* QuickTime 2.1 for Macintosh allows digital video compression cards to provide QuickTime with memory to store compressed video data, as it is read from the disk, during playback. If a compression card supports this feature, the demands on the processor's bus can be reduced by up to 50 percent, thus providing smoother video playback.

QuickTime 2.1 Availability

A package of QuickTime 2.1 software (including a new version of MoviePlayer and sample code illustrating sprite animation) is on the October 1995 System Software Developer CD (path—Dev.CD Oct 95 SSW:What's New?:New System Software Extensions:QuickTime 2.1 for Developers). You can also get QuickTime 2.1 and related files from the World Wide Web for free, starting at location http://quicktime.apple.com/. Users can also purchase the QuickTime 2.1 for Macintosh run-time software for a nominal fee of $9.95 through ZiffNet and ZiffNet/Mac.

Apple and Digitool Ship Macintosh Common Lisp 3.0

Apple Computer, Inc., and Digitool, Inc., recently announced availability of the Macintosh Common Lisp (MCL) 3.0 programming language and associated products. An enhanced version for Motorola 680x0-based Apple Macintosh computers, MCL 3.0 is the first product resulting from last November's technology development agreement between Apple and Digitool. MCL 3.0 is the first milestone leading to Digitool's Power Macintosh "native" version, expected in early 1996.

Common Lisp is an advanced object-oriented dynamic programming language widely used in industry, education, and research. Its flexibility and high-level access to the Macintosh user interface make it a useful tool for in-house developers, consultants, and others who are called on to create and modify software solutions quickly. MCL 3.0 is the first MCL version to support concurrently running multiple processes within a program.

Several new features of MCL 3.0 were included to meet customer requests: smaller deliverable applications, an MCL run-time compiler, and more affordable pricing. MCL 3.0 creates fully customized applications called Lisp Development Systems, which may be distributed to other MCL users in a site-license agreement or packaged with copies of MCL for commercial distribution. In addition, freely distributable, stand-alone applications may be produced with MCL Redistribution Kits; stand-alone applications now can occupy 450K–900K less disk space than their Lisp Development System counterparts produced in MCL 3.0.

MCL 3.0 can also be used to meet the demands of developers who require extensive network and multithreading interfaces. World Wide Web site development is one example of an application that takes advantage of MCL's strengths. The Artificial Intelligence Lab at M.I.T. successfully used MCL 3.0 for the Macintosh port of CL-HTTP, the lab's Web server, originally developed on Symbolics computers for government applications.

Pricing and Availability

MCL Version 3.0 is now available to companies and educational institutions for U.S. $595 and U.S. $475, respectively, through APDA, Apple's source for development tools and related programming products. For pricing on other MCL products, including a special student version, go to the Digitool Web page at location http://www.digitool.com/.

Apple Resumes Shipping PowerBook 5300

On September 25, Apple Computer, Inc., issued the following statement:

Apple Computer, Inc., today resumed shipments of the PowerBook 5300 series, and Apple USA also announced an immediate price reduction to U.S. dealers on purchases of the new PowerBook 5300 models. This reduction should result in an approximate street price drop of $100 (U.S.).

On September 14, Apple reported a safety problem with the lithium-ion battery packs in two early-production PowerBook 5300 units. Apple immediately halted shipments, and has contacted virtually all customers (fewer than 1,000 worldwide) who have received systems. Apple has since replaced all of the lithium-ion batteries in these PowerBook models with nickel-metal-hydride (NiMH) batteries, and has resumed shipments.

New PowerBook Prices

Model	Configuration	Price range
5300	Grayscale, 8/500 MB	$2,099-$2,199
5300cs	Dual-scan color, 8/500 MB	$2,699-$2,799
5300cs	Dual-scan color, 16/750 MB	$3,399-$3,599
5300c	Active-matrix color, 8/500 MB	$3,599-$3,799
5300c	Active-matrix color, 16/750 MB	$4,399-$4,599
5300ce	Active-matrix color, 32 MB/1.1 GB	$6,399-$6,699

There are no known safety issues surrounding NiMH batteries.

According to David Nagel, Apple's vice president for worldwide research and development: "The PowerBook 5300 is the most powerful PowerBook Apple has produced yet, and we're back in full production, using NiMH batteries. The PowerBook 5300 models are the first Apple notebook computers with PowerPC microprocessor technology. Using the energy-efficient PowerPC 603e, the 5300 series gives customers the best mobile computing solution in its class."

The new estimated street prices for PowerBook 5300 models, based on the price reduction to dealers, are expected to fall in the price ranges shown in the box on this page.

The prices shown are valid in the U.S. only, and may vary at the discretion of the dealer. Worldwide pricing may vary by region. Please contact your local dealer for specific pricing, configuration and availability information.

Apple, H-P Announce Macintosh Application Environment for Japanese Market

Apple Computer, Inc., and Hewlett-Packard Company recently announced that they are working together to create and distribute two Macintosh Application Environment (MAE) products for the Japanese market. These products will allow customers to run Macintosh productivity applications on their Hewlett-Packard and Sun UNIX workstations.

MAE is a software implementation of the Macintosh environment that uses 68LC040 emulation and "native" extensions to provide a virtual Macintosh environment in an X Window. In addition, MAE provides a high level of integration between the Macintosh environment and the UNIX operating system, enabling users to manipulate UNIX files and execute UNIX applications from the Macintosh environment. MAE 2.0 also adds important features such as AppleTalk networking, performance enhancements, MacTCP, and sound. Using MAE 2.0, Japanese customers will be able to run "off-the-shelf" Mac OS 680x0 applications, thus increasing their productivity on their H-P and Sun UNIX workstations without the expense of buying a separate Macintosh computer.

There are two MAE products for the Japanese market. The first product, MAE 2.0 with Japanese Language Kit, gives users the ability to run Japanese-language Macintosh applications in an English-language Macintosh environment. The second product, Nihongo MAE 2.0, gives users the ability to run Japanese applications in a fully localized Japanese System 7.1 Macintosh environment. Hewlett-Packard Japan will sell and support both products in Japan and key international markets.

Hewlett-Packard Japan will also provide MAE 2.0 Plus for HP-UX on Hewlett-Packard workstations. This product will combine MAE 2.0 with Japanese Language Kit and LaserVIEW, which can easily and quickly retrieve electronic dictionaries from a hard disk or CD-ROM while online.

MAE 2.0 Plus for HP-UX (including MAE 2.0 with Japanese Language Kit) is scheduled to be available by the end of September 1995, and MAE 2.0 with Japanese Language Kit for Solaris is scheduled to be available by the end of October 1995. Both Nihongo MAE 2.0 for HP-UX and Nihongo MAE 2.0 for Solaris are scheduled to be available in the first half of 1996.

Copland to Be Supplied in "Developer Releases"

Here is a statement from the Copland Development Team on the progress of the next major revision to the Mac OS, codenamed Copland:

The Copland development efforts will be moving away from the traditional alpha, beta, golden master cycle to a developer release cycle—a new model that we have successfully used during the development of OpenDoc. The goal of this new system is to deliver a series of progressively more complete developer releases until we have met our stringent compatibility, stability, and performance goals—at which point, we declare the product golden master and ship it.

This fall we will have a release for software-tools developers, so they can begin to develop the compilers and other software development tools application developers need to be productive with Copland. We believe this approach will help ensure that development tools are available by the time Copland is ready to be seeded to our developer base. We will be doing a widespread developer release seed early in the spring of 1996.

At last May's Worldwide Developers Conference, we announced that we would be providing more information later in 1995 on the seeding of Copland developer releases. In this article, we are announcing the first developer releases and overall release philosophy. In upcoming issues of *Apple Directions*, we'll keep you up-to-date on the status of the Copland developer releases and upcoming developer events.

Copland is the most complex software engineering effort that Apple has ever undertaken, and we realize that the golden master date is difficult to predict with a high degree of accuracy at this point. Once a couple of developer releases are behind us and we've had a chance to get more detailed feedback from developers and customers, we'll be in a position to make commitments about further developer releases and final product availability.

We regret these delays, but we're excited about Copland and our highest priority is to do it right. We want you to be able to do it right, too, and we'll keep you posted about how you can get ready for Copland through *Apple Directions*, *develop* magazine, the Mac OS Web site (http://www.austin.apple.com/macos), eWorld, AppleLink, and other communications vehicles, as well. ♣

Technology

CD Highlights

Tool Chest Edition, November 1995

This month, the Development Tools and Languages folder takes a rest; there were no updates to it since the last Tool Chest edition, and its absence gives us room to include three new versions of Korean system software that were too late for last month's System Software edition.

Two SDKs get their run times updated this month: Apple Guide advances to version 2.0, PlainTalk to 1.4.1, and QuickDraw 3D to 1.0.2. Updates to the SDKs themselves will follow in the near future. And, in addition to updates to the DTS QuickTime Utilities, HyperCard Player 2.3, MacsBug 6.5.2, the PowerTalk/PowerShare Products Guide, and Snippets, here are this month's new packages.

ADB Key Spy

This package provides something of a replacement for GetKeys by maintaining a key map reflecting the state of each keyboard attached to the Macintosh computer through the Apple Desktop Bus.

The package is intended to solve a problem a game developer was having with the adjustable keyboard, which appears on the ADB as two keyboards. It was impossible for said developer to allow users to steer on the numeric key pad and fire with the Space bar.

Drive Setup 1.0.2

Drive Setup is a program that lets you partition, initialize, and update fixed and removable disks on Power Macintosh computers.

Guide to Macintosh Development Tools

This document discusses the advantages of developing on the Macintosh platform and provides you with an overview of Macintosh development tools. Whether you're an application developer, solution integrator, client/server developer, or multimedia author, this guide provides you with information to help you make informed decisions on tool purchases—decisions that will ultimately result in faster development and better software.

Tool Chest Edition

Macintosh PowerBook Qualified Program

This is the program guide for the new Macintosh PowerBook Qualified logo licensing program. This is technical, legal, and marketing documentation in Adobe Acrobat format.

The kit includes

- a guide to the Macintosh PowerBook Qualified Program
- program technical specifications
- program pretesting guidelines
- legal documentation for licensing and program participation

PowerPC Documentation

This folder contains documents about programming for the PowerPC processor. These documents, referenced in *Inside Macintosh: PowerPC System Software,* have previously only been available on E.T.O. The documents titles are *Debugging Optimized Code, Libraries* and *OOP, Making the Leap to PowerPC, and Moving Your Source to PowerPC.*

ProjectDrag 1.1b8

ProjectDrag is free drag-and-drop source control software, based on SourceServer from Apple Computer, Inc. SourceServer is an Apple event–based subset of the MPW Shell that implements the Projector source control commands. See the column in *develop* Issue 23 (pages 72–76) for information on SourceServer and ProjectDrag basics.

Spain Distribution Guide

Apple Computer España has prepared this guide to help third parties whose products are not yet distributed in Spain.

The guide includes general information about Spain and about the position of Apple Computer España in the information technology sector. You'll also find all the information you need to start marketing your products in Spain, as well as the names of relevant companies, distributors, magazines, and so on.

VersionEdit 1.1

VersionEdit is a software development tool that automatically creates and updates product version information throughout the life of a software product.

As an alternative to using ResEdit or Rez, VersionEdit can be used to create and

Inside This Section

OpenDoc Human Interface FAQs 14

Optimizing PowerPC Programs 17

CD Highlights

intelligently increment 'vers' type resources that adhere to Apple's version guidelines. VersionEdit also allows developers to encapsulate additional information about the product and their company within the application, making it available for use by other utilities. See the About VersionEdit file for details.

Please note that 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.

Coming Next Month
Adobe Acrobat—the story continues. . . .

Alex Dosher
Developer CD Leader

OpenDoc Human Interface FAQs

Designing for Users With Disabilities

By Kerry Ortega and Geoff Schuller, OpenDoc Human Interface Team

Gary Moulton and Peter Green, Worldwide Disability Solutions Group

This FAQ is devoted to the discussion of how OpenDoc's technologies and the potential for third-party disability solution parts relate to users who have "special needs."

Q: How likely is it that someone with a disability might want to use one of the parts I've developed, and why should I spend extra energy making sure my parts provide universal access?

A: Worldwide, 550 million individuals have disabilities. These individuals, because of accident, illness, congenital condition, or aging, have reduced visual, hearing, physical, or cognitive/speech abilities. Such disabilities prevent individuals from using standard microcomputers, operating systems, and application software. Also, additional millions of users have RSI (repetitive strain injuries) and can benefit from many of the solutions aimed at people with disabilities.

Fortunately, Apple has designed Macintosh hardware and system software to make Macintosh computers more adaptable those with special needs. For example, the Apple Desktop Bus makes it easy for people to add alternate input and output devices, and the Easy Access control panel is just one of many pieces of system software that can help people with various disabilities. By making your OpenDoc parts universally accessible, you allow more users to use your parts, and that's good for everybody.

Apple Computer, Inc., has a ten-year history of making personal computer hardware and software accessible to people with disabilities. Apple's efforts in this area began before laws and regulations were implemented to mandate access (for example, the Rehabilitation Act Amendments of 1992). As a result, Apple products are fully and easily accessible to individuals with disabilities, and current data indicates that Apple's disability market share is 10 million dollars. This means that there's a lot of market potential for your OpenDoc part, if it is developed with universal access in mind.

Q: What can I do to make sure the OpenDoc parts I develop provide universal access?

A: Just as city developers have an obligation to permit equal access through curb cuts, software designers have an obligation to create "electronic curb cuts." City planners have discovered that curb cuts are much less expensive to install if they are planned for before the sidewalks are built. Similarly, the earlier you plan for disability access, the less expensive it will be for you to implement—the expense goes way up if you have to retrofit disability access into an existing product.

The general rule when thinking about making software universally accessible is to provide alternative ways of accessing the information. So when an audio cue is used to relay information, it's helpful if you offer a video cue as well (or at least provide a way to choose between the two). When using text, include options that will read that text out loud, enlarge the font size, or switch colors. Easy Access, for example, is the best-known software "electronic curb cut," one that in effect gives an alternative way of selecting multiple keys in the Mac OS. Easy Access is a prime example of a universal access software feature—a built-in accommodation for people who are unable to do a particular task in the standard way.

In addition to the access features mentioned earlier, two other exemplary "electronic curb-cuts" are QuickTime closed captioning and the text-speaking ability of SimpleText. Built into QuickTime is a text track that can be used for creating closed-captioned versions of QuickTime movies—a feature that is useful for deaf and hearing-impaired users. Also, SimpleText includes an option that uses the PlainTalk text-to-speech service to "read out loud" any text in the word processor; this is useful not only for people who are visually impaired, but also for others who have trouble reading—for example, people with dyslexia.

Even if you don't have the resources to add disability-access features to your part (or any other product), it's important for you to figure out what needs to be done. With that knowledge in hand, you may be able to arrange for a third-party company to provide the needed access solution. This solution is possible only if you plan for it while you are designing your product.

Q: What kinds of parts would be useful to build for people with disabilities?

A: There are a lot of great opportunities for developers who want to create parts for people with RSI or various kinds of disabilities. These opportunities fall into two main categories: parts that help people write or draw, and parts that help people navigate between parts or documents.

For example, a "writing assistant" part could add word prediction and suggestion capabilities to an OpenDoc text processor part. These capabilities could help people with physical disabilities increase their typing productivity, and could help people with cognitive disabilities improve the quality of their written work. A part like this could "learn" new words as they are typed and immediately begin using them in the prediction process.

Another useful part would be a Braille part. It could be used to convert text into Grade II Braille, a version of Braille in which words and combinations of letters appear in a condensed form. Once a document has been translated to Braille, it could be printed out on a Braille printer/embosser. A part such as a Braille translator would help blind and sighted people work together in any type of environment.

Another part that would be enormously helpful for visually impaired users would be one that reads text out loud. An obvious use would be to read the contents of an entire document, but a part like this could also read characters as they are typed, words as they are completed, or sentences as they are completed. This kind of part could also be used by speech-impaired people who want to use the Macintosh to produce audible communication that can be heard by others.

Another good opportunity for a disability solution part is one that would serve as a replacement for the standard keyboard and mouse. With a part like this, a user with a severe physical disability (for example, people with few muscles under voluntary control, who use just one hand to type, or who must type with their feet) could create content and navigate between the parts in a document. This part could use a scanning sequence—that is, a sequential display of choices that allows users to make their selection by tapping a switch—to allow the user to draw pictures, open menus, manipulate icons, and do many other tasks using an input device as simple as a single switch (for example, a "mouse stick").

Q: What is OpenDoc doing to help people with disabilities?

A: OpenDoc allows the user to perform many operations without a mouse. For example, OpenDoc allows users to copy, cut, and paste icons using the keystroke equivalents assigned to menu items. This greatly benefits those who use alternate input devices and "software keyboards." Also, because OpenDoc has provided keyboard equivalents for commands like Copy and Paste, it will be possible for someone to develop a "software keyboard" part that could easily communicate with OpenDoc, thus enabling users of these input devices to use these very important functions. (Software keyboards are on-screen representations of keyboards that people incapable of using a physical keyboard can control by some other means—a head movement, for example.)

In the Finder, there's currently no way to delete a document except to drag it to the Trash, which is pretty difficult for people who have problems with using the mouse. In OpenDoc, we've added a Delete Document command to the Document menu, which makes deleting files a little easier for some of our users, and makes it possible for "software keyboards" to perform this function. And because the Delete Document command saves the contents of the current document, closes it, and moves it to the Trash (without actually deleting it from the hard disk), there's an easy way to recover the file, if necessary.

In future versions of OpenDoc, we'd like to add a way to navigate through the embedded part's hierarchy using the keyboard. One design we're considering uses the Shift and arrow keys. The Shift–down-arrow combination would navigate down into a selected part (that is, activate it), and the Shift–up-arrow combination would navigate up into the containing part. In order to navigate down through the part hierarchy, OpenDoc would have to provide a way to select embedded parts through the keyboard. We appreciate any comments you have on keyboard navigation (send to OpenDoc-Interest@CI Labs.org).

Q: What is currently done on today's Macintosh computer to support people with disabilities?

A: The Macintosh includes certain built-in access features that are specifically designed for individuals with disabilities, but used by everyone. These features include

- CloseView (which enlarges the screen image up to 16 times)
- visual beep (which blinks the menu bar to indicate a "beep")
- selectable key repeat (a feature of the Keyboard control panel that helps users with motor-control problems)
- floppy disk access (Macintosh computers do not require the user to lift and lower a latch to insert a floppy disk; disk ejection can be initiated in several ways; and ejected disks are positioned to be easily grasped)
- Easy Access (which substitutes the keyboard for the mouse and makes it easier for users to press multiple-key combinations)
- SlowKeys (which lengthens the time a key must be held down to register as a valid keystroke, thus reducing mistyping by people who have motor-control problems)

These features come with the Macintosh when you buy it. Apple is continually working to ensure that additional access features are included in its new generations of computers.

Q: Where can I find more information on the Worldwide Disability Solutions Group?

A: The Disability Connection on eWorld (shortcut: DIS) contains information relating to the field of Macintosh assistive technology solutions. Refer to this area to learn about related events and state-of-the-art products used to customize Apple computers for use by individuals with disabilities. Apple has a disability solutions page on the World Wide Web, located at http://www.apple.com/disability/welcome.html. You can also contact the group at the Internet address applewdsg@eworld.com, or by mail at the following address:

Apple Computer, Inc.
Worldwide Disability Solutions Group
1 Infinite Loop, M/S 38-DS
Cupertino, CA 95014 ♣

Optimizing PowerPC Programs

By Gregg Williams, Apple Directions *staff*

So you've gotten your application to run native on Power Macintosh computers and it's perfect, right? Well, nothing's ever perfect—especially software.

A good number of developers have found out that, with a little work, they can make their applications run considerably faster than they currently do. Also, new Power Macintosh computers that use second-generation PowerPC processors (the Power Macintosh 9500, for example) may even run software more slowly than earlier Power Macintosh computers do—which will not make your customer think kindly of you and your product. Add to that the need for the Mac OS platform to be as good as possible in a highly competitive market, and you have a number of compelling reasons to consider optimizing your application.

You may not be able to define optimization, but you should know it when you see it—your customers are happier with your application. In some cases, you actually make your code run faster; in others, you streamline your application so it takes less time for your customers to get their work done. And even if they don't appreciate what you've done, at least they'll be less likely to complain or switch to the competition.

Optimizing is, at best, an art, and it's hard to generalize from individual examples—but examples are all we have. The recommendations made in this article come from Apple engineers in their work with various third-party developers (most of whom have since released new, faster applications).

What I've tried to do in this article is to group the data from different optimization efforts into meaningful categories. At first glance, the categories may seem too general, and the examples too specific, to be of practical use. But optimization depends as much on intuition as it does on cut-and-dried procedures, so it's quite possible that one of the examples you read may spark some insight that will help you optimize your application—even though the example itself may not directly apply to your situation.

Overview

There's no substitute for reading this entire article—optimization is in the details. But if you don't want to do that right now, here are some top-line thoughts from the rest of this article.

- You should define your goals before you start trying to optimize your application. Just saying "I want to make my code run faster" isn't enough.
- You can't optimize your whole application. You must profile your code to see where it's spending most of its time and optimize those routines.
- The new PowerPC processors (including the PowerPC 603, 603e, 604, and 620 processors) do things differently from the PowerPC 601 processor. If you don't deal with data-alignment issues, your application may run more slowly on new and future Power Macintosh models that use these processors than it does on the older Power Macintosh models that use the PowerPC 601 processor.
- By far, the most common source of optimization is finding unnecessary operations that your application currently performs and eliminating them. See the section titled "Unnecessary Operations" for examples.
- Numeric calculations may be the source of some performance improvements. You can definitely speed up execution by replacing FixMath fixed-point arithmetic routines with floating-point calculations in your program; your compiler will compile these to native PowerPC floating-point operations. It's also possible that, running on a fast processor, your code can calculate a value from a formula faster than it can look up a precalculated value from a table (a technique that is often found in existing code).
- You can sometimes speed up graphic operations by working on four 8-bit pixels at a time.
- It's also possible to improve performance by optimizing an algorithm or by substituting a smaller custom routine for a general-purpose system routine.
- One of the most important optimizations you can make is to increase the size of your application's I/O buffers to at least 16K, though 64K may result in further improvement.
- If you have access to the Internet and a World Wide Web browser, you should go to the site at location http://www.info.apple.com/ppc/OptiPages/main.html. This Web page is a clearinghouse for various articles, reference documents, and software tools relating to optimization.

Enough preliminaries—let's get started!

Define Your Goals

As the saying goes, "If you don't know where you're going, it doesn't matter what direction you start in." It's the same way with trying to optimize your application—if you don't set a goal for optimization and a plan for achieving it, you'll never know when your job's done, and you may spend more effort and still achieve fewer results.

Here are some basic ground rules for optimization. You probably already know them, but it doesn't hurt to see them in print:

- Decide on a specific goal for optimization and determine how you are going to measure it—otherwise, you won't know if you have been successful.

It's not enough to say, "Well, I just want it to run faster, that's all." And before you can set a specific goal—"I want to cut the load time in half and make scrolling at least 25 percent faster," for example—you have to know what you want to improve. That means measuring current performance—yourself, if necessary, but preferably with typical end-users. Ask the question, "Where does it seem slow?" Also test competing products and make sure you don't lose big to them.

- It doesn't pay to tune all of your code—programs are just too big these days. Also, optimization is hard work, and it doesn't make sense to optimize code that rarely gets called. Use profiling tools to find the critical areas and bottlenecks in your application, then focus on optimizing them.
- Macintosh computers spend their time executing two kinds of code—the code you've written, and the code that Apple has written (system routines). You can only optimize code that you've written. If you come across system routines that take a lot of a program's execution time, you can't change that. What you can do, however, is see if you can make your code call such routines less frequently, or you may be able to substitute a special-purpose routine that does the job faster.

Use Performance Tools

Optimization is, at the highest level, very simple: Find out where

your code is spending its time, and change it to be more efficient. Here are a number of tools you can use to improve your code's performance.

- Every major programming environment includes some kind of profiler. In addition, the Macintosh Debugger (part of the E.T.O. CD, sold by APDA) includes an Adaptive Sampling Profiler that samples the processor's program counter to "measure" how often different sections of code execute.
- PowerTracer is an A-trap and PowerPC function tracing and timing tool. You can use it to generate detailed trace output information, including the execution architecture for each system function (that is, 680x0 or PowerPC), execution architecture for the caller of a system function, and interrupt information.
- IOTracer (version 6.0 is the current version) is a tool that helps you analyze System 7 file-system and disk I/O calls. You can use this tool to discover which file-system calls are being used and where your code is spending its time when it triggers the execution of file-system calls.
- 4PM is a performance monitoring tool. The PowerPC 604 processor includes new features that monitor the processor's performance (for example, the number of data cache misses). You can use the 4PM tool to control and access these features. (The PowerPC 603 and 603e processors don't have this performance-monitoring feature, but it will probably be incorporated into some or all of the future PowerPC processors.) Issue 24 of *develop* (due out in late November 1995) will include an article on how to use 4PM in its "Balance of Power" column.

You can find PowerTracer, IOTracer, and 4PM on the latest Tool Chest Developer CD. On the August 1995 Tool Chest CD, all three tools are at pathname DevCD Aug 95 TC:Tool Chest: Testing & Debugging:General tools:Tracer.

Optimizing for Post–PowerPC 601 Processors

The Power Macintosh 9500 is the first Power Macintosh computer that uses the PowerPC 604 processor, and future models will also be using this and other post–PowerPC 601 processors. The main reason for mentioning this is that these later PowerPC processors have some important architectural differences from the PowerPC 601. (For details, see "Preparing Your Code for Future PowerPC Processors" in the May 1995 issue of *Apple Directions*.)

By making some small but significant changes to your application, you can make it run faster on post–PowerPC 601 processors. In some cases, you're making the application actually run faster; in others, you're correcting a situation in which your application, because of the way it's coded, runs more slowly on these new processors than on a PowerPC 601.

Are the changes worth making? Here are the results that one developer got when optimizing a graphics application for the PowerPC 604 (numbers listed are "times faster than the fastest version running on a PowerPC 601"):

- Gaussian blur—2.0
- blend—1.5
- sizing down—1.3
- paint (along a path)—1.7

With that said, here are some examples of optimizations that resulted in applications running faster on a PowerPC 604 processor. The first example, which deals with misaligned data structures, is the most important one.

Misaligned Data Structures

When an application performs poorly on a post–PowerPC 601 machine, the problem is often improper data alignment. PowerPC processors need a different alignment than do 680x0 processors; data needs what is called "natural alignment"—that is, 2-byte quantities should be aligned on 2-byte boundaries, 4-byte quantities should be aligned on 4-byte boundaries, and so on.

Define your goals: Just saying "I want my code to run faster" isn't enough.

The problem for post–PowerPC 601 processors is that the PowerPC 601 processor is much more forgiving of misaligned data than the newer processors are. Some developers who converted existing 680x0 Macintosh applications to run on Power Macintosh computers simply stayed with the original code's 680x0 data alignment because, at the time, doing so incurred little or no performance degradation. Now, on Power Macintosh computers that use the newer processors, it does.

If an address is aligned on a 2-byte boundary, the penalty on a post–PowerPC 601 processor for an integer access is minimal, but floating-point and load/store multiple operations execute more slowly when their data is misaligned.

Because of data-alignment problems, the original Power Macintosh version of one high-end desktop publishing application ran much more slowly on a PowerPC 604 processor–based Power Macintosh than it did on one with the older PowerPC 601 processor (oops!). Once these problems were fixed, the PowerPC 604 version ran eight times faster than before, making it about 25 percent faster than the same program running on a PowerPC 601 processor–based Power Macintosh computer running at the same frequency.

Recompiling for Post–PowerPC 601 Processors

Just recompiling an existing Power Macintosh application for post–PowerPC 601 processors (done by setting the appropriate compiler switch) should improve performance modestly. One developer estimated that doing this alone would improve his graphics application's performance by about 10 percent.

Calculating a Formula Instead of Table Lookup

You should keep in mind one central fact: As processors get faster, the best way to get the job done may change. Doing a specified calculation as quickly as possible is a case in point. In the past, the fastest way to do a specified calculation has been, in many cases, to precalculate the result for a given range of inputs and do a table lookup.

As factors such as processor speed and architecture, cache size, and memory access time change, table lookup may no longer be the fastest solution. (For example, if the table is large enough, every table lookup will cause a cache miss, thus slowing the lookup process significantly.)

Apple engineers and third-party developers have found that, in some cases, it's faster to make the desired calculation directly

instead of doing a table lookup. (This technique is also more flexible and memory-efficient.) Depending on the circumstances, direct calculation may be the way to go—not only on the newer PowerPC processors, but also on the PowerPC 601 and even the Motorola 68040 processor.

However, the developer of one graphics application found that table lookup was still more efficient than direct formula calculation for two operations:

- LAB-to-CMYK conversion
- CMYK-to-RGB conversion

The reason for this is that the conversions used complicated formulas that still took more time to calculate on a PowerPC 604 processor than a simple table lookup would have.

Another developer's experience confirmed the previous example. On the PowerPC 604, multiplication operations are "cheaper" than on a PowerPC 601, and true multiplication is often faster than a table lookup. In the example given by this developer's graphics application, a table lookup (which was faster on a PowerPC 601) was 25 percent slower on a PowerPC 604 than the equivalent multiplication operation.

Unnecessary Operations

Of all the optimizations recorded, the category of "unnecessary operations" had the most entries. Let's face it—programming is not an exact science, and today's programs (which are unimaginably huge by the standards of ten years ago) "grow" organically. As a result, code may contain duplicated or unnecessary operations. Going one level of subtlety deeper, one perfectly reasonable routine may do something unexpected in conjunction with another perfectly reasonable routine.

It's pretty likely that some unnecessary operations are in your program. How do you find them? Use profilers and examine the results extremely carefully. Scrutinize every routine that consumes, say, over 2 percent of total execution time or time spent in the Toolbox. Ask yourself, "Is it reasonable for that routine to spend that much execution time?" If you're not entirely sure the answer is yes, investigate further.

Routines Called Too Many Times. Some routines may be using excessive amounts of execution time because they are called a large number of times. Others do so because they are implemented as non-native (680x0) routines and run slowly because, on a Power Macintosh, they run in emulation.

Remember also that a Power Macintosh computer incurs a mixed-mode overhead of about 50 microseconds when it switches from executing native PowerPC code to 680x0 code or vice versa. At one point, TickCount was emulated, and one developer called it from within a calculation loop of PowerPC code; he found that the mode switching caused execution to be three times slower.

Here are some other examples:

- A developer found that his 3D animation application spent significant time in the routine DeleteMenuItem; for example, during a file-open operation, DeleteMenuItem is called over 65,000 times. Since this file-open operation is known to spend well over 90 percent of its time in the Toolbox, removing extraneous DeleteMenuItem calls could produce some significant performance gains. Similarly, in this application, DeleteMenuItem consumed 45 percent of the time spent in the Toolbox during close-, open-, and new-file operations and 39 percent of the Toolbox time spent in closing a file.

Since DeleteMenuItem is not native, it executes slowly and should not be used excessively. It's possible that calls to DeleteMenuItem might be moved out of some or all of the loops they're currently found in.

- A developer found that his graphics application spent significant time in FindWindow and HiliteControl, and he couldn't account for why the given code might be calling those routines. Since those routines are not native, they execute slowly, about 16 and 21 milliseconds, respectively. If those routines are, in fact, not needed, removing them could produce some modest performance gains. Other possibly extraneous Toolbox routines that this application called while rendering an image include TickCount, EventAvail, and HGetState.
- One graphics application was calling SetCursor about 40 times per second, which is more than is needed. By reducing the frequency of the call, the developer was able to make a modest (several percent) performance improvement.
- Profiling the time spent in the Toolbox during a file-open operation within a page-layout application, a developer found that PBHGetFInfoSync and PBGetCatInfoSync were called 138 times and 230 times, respectively. The developer was not sure what these calls were doing, but they were taking about 6.3 percent of the time spent in the Toolbox during a file-open operation. This was a possible source of a modest performance increase.
- One page-layout application was calling several routines (including SetPort and GetPort) over 5,000 times during a text search-and-replace operation. The developer estimated that the application would likely see a 20 to 30 percent performance improvement by eliminating superfluous calls.
- Many applications call EventAvail too often, and you may gain a bit of performance by limiting the number of times it gets called. You can probably avoid calling EventAvail if fewer than 4–6 ticks have elapsed since the last time you called it. User studies show that humans begin to perceive intervals in the range of 1/10 second (6 ticks) to 1/15 second (4 ticks) as instantaneous.

WaitNextEvent. Another place where you may be able to increase your performance is by intelligently limiting your use of WaitNextEvent during lengthy routines (any routine that takes more than a fraction of a second).

Resources

- "Preparing Your Code for Future PowerPC Processors," *Apple Directions,* May 1995, page 14. Also available on the World Wide Web at http://www.info.apple.com/dev/appledirections/may95/techcode.html.
- Tech Note PT38—"PowerPC Compatibility and Performance Issues." Available on the Developer CD Series. Also available on the World Wide Web at http://www.info.apple.com/dev/technotes/Platforms_&_Tools/pt_38.html.
- World Wide Web location http://www.info.apple.com/ppc/OptiPages/main.html contains information on porting programs to the PowerPC processor and optimizing PowerPC code.
- *develop* "Balance of Power" column: Issues 18 (page 55), 19 (page 17), and 24 (forthcoming).

In the past, developers often called WaitNextEvent after a fixed amount of code—which, on slower Macintosh computers, took no longer than (for example) 1/15 second. However, as computers got faster, that fixed amount of code took less time to execute, and WaitNextEvent was being called too often.

The solution is to have your routine call WaitNextEvent only after a reasonable amount of time has elapsed. Here's what one Apple engineer recommends: Have your routine periodically check a timer that is set for *X* ticks and call WaitNextEvent when the timer decrements to 0. What is *X*? Set *X* to a conservative value (4 ticks, for example). Then have your code check the value returned by WaitNextEvent. If the result indicates a null event (that is, no other process needs the processor), then increase the value of *X*. If the result indicates a non-null event, decrease the value of *X*. This maximizes the amount of time your routine spends on itself while keeping the computer responsive when the load on the processor increases.

HLock/HUnlock and HGetState/HSetState. Several developers found optimizations related to these routines:

- Profiling the time spent in the Toolbox during a file-open operation within a page-layout application, a developer found that HLock and HUnlock repeatedly locked and unlocked the same block of memory (thought to be cached font information). Together, those two routines took up 18 percent of the time used by the Toolbox, so decreasing the number of times these two routines get called could increase the speed of a file-open operation by up to 18 percent.

In a similar way, HLock and HUnlock took up 18 percent of the time used by the Toolbox to import a text file, 28 percent of the time to do a text-placement operation, and 51 percent of the time during a text search-and-replace operation.

- One graphics application spent 7 to 16 percent of its total execution time executing HSetState, HGetState, and HLock. The routines were being called between 500 and 1,000 times per second. Depending on the circumstances, it might be possible to increase performance by locking the handles once for long operations.

Operations That Don't Make Sense. The fact that your code doesn't crash doesn't mean that everything about it is right. By profiling your code and thoughtfully analyzing the results, you may find code whose presence "just doesn't make sense."

- In the case of one graphics application, the developer profiled his text-drawing code and found that it spent 35 percent of its time in the CharWidth function. This was eventually traced to a bug in Adobe Type Manager (which was subsequently fixed).
- One page-layout application spent 15 percent of its Toolbox time during a text-file-import operation executing GetResource. This is suspicious because the file being imported was text-only. Also, GetEOF was called 109 times, for 7.8 percent of the time spent in the Toolbox during this operation. In a scrolling operation, DrawGrowIcon was called 77 times, for 4.8 percent of the time spent in the Toolbox during that operation. All these routines are candidates for further study; decreasing their usage would speed the text-file-import operation by up to 10 percent.

Out-and-Out Mistakes. It's interesting to note that all but the first example here are directly involved in drawing within graphics-intensive applications.

The fact that your code doesn't crash doesn't mean that everything about it is right.

- Unnecessary operations can bring an application to its knees. A word-processing application was suffering from sluggish text entry. Upon examining the application's code, the developers discovered that the status bar at the bottom of the screen was being updated on every "null" system event, and this updating was occurring even when the status bar's contents hadn't changed. Worse yet, the code was updating the status bar twice per null event! Correcting these problems made text entry 200 to 300 percent faster and greatly improved the responsiveness of the product.
- One developer found, during profiling, that his graphics application was drawing each text string twice. Needless to say, eliminating the extra draw operation improved performance.
- One graphics application called PenPat twice for every call to PenPixPat; this may have been a programming error.
- In a graphics application, a developer found a large number of unneeded redraw operations in his code. In an image-playback operation, eliminating these unneeded redraws decreased the amount of time spent in the Toolbox by about 7 percent. In an image-scroll operation, eliminating the extra redraws decreased the amount of time spent in the Toolbox by about 77 percent, making the operation four times faster.
- In one graphics application, a developer estimated that about 70 percent of the routines called during the scrolling of an image were unnecessary.

Caching Invariant Values. So you found a routine that's eating up a lot of execution time within a loop? Check to see if the routine's only effect is to return a value. If so, and if that value remains invariant part or all of the time, perhaps you can find a way to cache that value and use it instead of calling the function again. Here are some examples, both of which involve system routines that are not PowerPC processor–native:

- A developer found that his graphics application called GetGWorldPixMap as many as 3,000 times per second. Since GetGWorldPixMap is not native, each execution was taking about 86 microseconds and, because of this, was taking about 25 percent of a task's total execution time. The developer was able to speed up this application by caching and reusing the value returned by GetGWorldPixMap, which did not change between calls. Two other often-called Toolbox routines whose values did not change between calls were FrontWindow and GetPixBaseAddr.
- A page-layout application spent 23 percent of its time during a text-scrolling operation in FontMetrics. (FontMetrics took a large fraction of the time to execute because it's not currently implemented as PowerPC code and had to be emulated.) Since the developer knew the value returned by FontMetrics didn't change during this operation, he instead cached the value and used it during the

scroll, for a significant increase in performance.

Miscellaneous Examples. What can I say? These examples don't fit anywhere else, but they may trigger an insight in you about your code.

- Some operations are called in loops but need not be. By calling them only when necessary, you can increase performance somewhat. (This is especially true of routines that are not PowerPC processor–native and so execute slowly.) A developer found this to be true in his page-layout application.

For example, the non-native ShowControl routine took 15 percent of the time spent in the Toolbox during a text-placement operation. According to the developer, "I would recommend that ShowControl not be done until the whole document has been loaded."

The developer also found that the non-native DrawControls routine took 9 percent of the time spent in the Toolbox during a scrolling operation. He said that DrawControls could be taken out of the scrolling loop, for corresponding savings in execution time.

- Be on the lookout for extraneous Toolbox calls. By finding and removing such calls, one developer was able to increase the performance of his word processor's search-and-replace operations by 25 to 35 percent and certain graphics operations by 30 to 40 percent.
- A developer was able to speed text scrolling within a page-layout program by about 20 percent by not updating the scroll bar's maximum value (which doesn't change during the scroll operation) and by using a better algorithm to redraw the vertical ruler.
- An image-defringe operation within a graphics application modified the destination only if a given condition was true; however, it read the value that the destination was to be set to from memory, whether the destination was actually to be modified or not. The developer sped this operation up by not doing the read operation if the given condition was false—that is, if the destination *wasn't* going to be changed. This improved performance by about 12 percent.

Numeric Issues

You can use several techniques to potentially speed up a numeric operation. (Naturally, you would bother to do this only if you had already found out that your application spent a significant amount of time in the numeric operation.)

Consider Using Floating-Point. Remember, now that computers are much faster than they used to be (especially Power Macintosh computers), the way you used to get a numeric calculation done may no longer be the best way. Here are two examples:

- Check your application (especially if it has source code that predates 68040 processor–based Macintosh computers) to see if it relies heavily on FixMath routines. On both the 68040 and all PowerPC processors, floating-point operations are usually faster than FixMath routines. On the PowerPC processor, the fpdiv instruction is 22.7 times faster than the FixDiv utility routine, and the fpmul instruction is 8.67 times faster than the FixMul utility routine. On the Macintosh Quadra 950, fpdiv is 10.6 times faster than FixDiv, and fpmul is 5.93 times faster than FixMul.

The developer of a graphics application estimated that 1/3 to 1/2 the time spent drawing with paintbrushes was spent in the FixMath library, so converting to floating-point math would significantly speed up the application.

- You should also consider substituting a direct calculation for a precomputed table lookup. See the earlier section "Calculating a Formula Instead of Table Lookup" (page 16) for details.
- In a generalized scaling operation within a graphics application, a developer was able to speed the routine up by breaking the generalized operation (which could require division by various numbers) into a switch statement that handled different divisors differently. Cases involving the divisors of 2, 4, and 8 used shift operations for maximum speed, and cases involving divisors of 3, 5, 7, and 9 used special routines that used multiplication, addition, and shift operations to increase the speed of the computation. He reported that benchmarks for the cases of 2, 4, and 8 showed that "the speed-up was pretty good."
- In a paintbrush operation, a graphics application developer was able to save some time by substituting an unsigned multiplication operation for a signed multiplication operation. (He knew that signed multiplication was not needed in that situation.)

Loop Unrolling in Graphics Applications

A developer optimizing a graphics application achieved a moderate performance improvement by working on four 8-bit pixels at a time. A blend routine that worked on one 8-bit pixel value at a time was speeded up by about 30 percent by unrolling the loop to work on four pixels at a time, as a 32-bit value. One factor that contributed to the speed-up was that, for this blend effect, the routine had to check a mask value for each pixel. Working four pixels at a time, the new routines could often skip an entire four-pixel loop when the mask value was 0, thereby saving three compare instructions.

The developer also improved a paintbrush operation by 21 percent by unrolling a loop to work on four pixels at a time. He said this made painting seem "much more interactive than before."

Optimizing Your Routines

Of course, one way to improve a routine's performance (again, it should be a routine that's taking a noticeable fraction of your application's execution time) is to rewrite it! Sometimes you do this by substituting a faster algorithm. In other cases, you do it by getting the work done in a different way. Here are some examples:

- By rewriting existing code and using more efficient algorithms, you can often make significant improvements in a program's execution. For example, a developer working on a graphics application got a Gaussian blur routine to execute up to 60 percent faster.
- While drawing a complex shape, a graphics application spent about 5 percent of its Toolbox time calling TickCount, which it called about 5,000 times per second. By substituting LMGetTicks, which takes about half as much time to execute, you should be able to get a small (around 2.5 percent) speed increase.
- Work smarter, not harder! A company found that its program (which has a spell-checking component) ran 13 times faster on a Pentium system when compared with its PowerPC processor version. By checking with a developer on the PC side of the company (the PC and Macintosh versions of the program had been developed independently), the Macintosh developer found that the PC version of the spell-checking code ran faster because it worked differently. It marked checked material as "clean" and skipped over it if the user asked for another spell-check operation. Naturally, the

Macintosh version of the program will run much faster once it uses the same algorithm as the PC version.

Substituting Custom Routines

When an application is spending considerable time in one system routine, consider whether substituting a custom routine might improve performance. This is most likely to be the case when

- the system routine is running in emulation (and your substituted routine would, of course, be native code)
- the system routine is very general-purpose, and the task you need accomplished is very narrowly defined (and thus can be a shorter, more compact function)

Here are some examples:

- In one high-end, object-based graphics application, a developer found that the code for a "select all" operation spent 60 percent of its time in the CopyBits routine. Further examination revealed that the code was using CopyBits to draw four "handles" for each of (potentially) thousands of objects. CopyBits was being used primarily to "stretch" an 8-by-8-by-1-bit bitmap to the current screen depth. Because CopyBits is so versatile, it is of necessity not the most efficient routine for the task. Since this drawing function is called so many times, it made sense to create a finely tuned, single-purpose drawing routine to substitute for CopyBits. Doing so made the "select all" operation two to three times faster.
- A numeric-intensive application, it turns out, was spending about 15 percent of its time checking for the Command-period keystroke. By using lower-level operating system routines to do this checking, this application sped up its calculations by about 10 percent.

Compiler Issues

- Be aware that the fastest compiler isn't necessarily the best compiler for your final build. One developer used the Metrowerks MPW compiler because of its speed. The MrC and Motorola PowerPC compilers deliver better final code, so you may want to use your favorite compiler for development but another compiler—one that generates more optimized code—for your final build.
- If you have decided to optimize your code for the PowerPC 604 processor (or any other post–PowerPC 601 processor), remember to set the PowerPC 604 compiler switch (as mentioned in the earlier section "Recompiling for Post–PowerPC 601 Processors").

In most cases, doing so will not have any appreciable effect on the same code running on a PowerPC 601 processor–based Power Macintosh. If it does, however, you have to decide who your most important customers are and which Macintosh models they're most likely to be using. The answers to those questions will help you decide how to optimize your code.

Miscellaneous Issues

Though these examples are at the end of this document, don't overlook the first example below—changing the size of your application's I/O buffers, which is easy and can net you considerable improvement. Apple engineers recommend that you increase your I/O buffers to 16K, if possible; you may want to try a 64K buffer to see if that improves performance.

- In a graphics application, a developer found that changing the application's I/O buffers from 2K to 8K improved file-open operations by 20 to 25 percent.
- CopyBits can degrade an application's performance with offscreen bitmaps if not used properly. In one graphics application, CopyBits was being used to redraw a finished offscreen bitmap into an onscreen window. Unfortunately (and this was not obvious at first), the color table for the offscreen buffer didn't match the color table for the onscreen window, forcing a slow subsection of the CopyBits routine to execute. By making the color tables of the offscreen bitmap and the onscreen window the same, CopyBits ran two to three times faster, which increased the application's performance by 50 to 80 percent in a number of areas.
- Depending on your situation, this information may help you if you're trying to decrease your application's startup time. In a certain graphics application, startup time is heavily influenced by resource-related routines, which themselves spend much of their time reading from the disk. Here are some statistics that may provide you with some clues about how to reduce the startup time of your application:

— Looking at all Toolbox calls, instead of ones called directly from the application, we can see that 37.1 percent of total startup time is used by PBReadSync.

— 86 percent of the startup time is spent in Toolbox calls, with the remaining 14 percent spent in actual application code. Here is a list of all the Toolbox routines that consume more than 5 percent (each) of the total startup time:
LoadResource—9.5 percent
GetResource—8.1 percent
NewPtr—6.9 percent
Get1Resource—6.9 percent
Get1IndResource—6.4 percent

Note that all but one of them are resource-related.

- If you are not aware of the limitations of the profiling tool PowerTracer when you use it, you may interpret its output incorrectly. Because it puts a header and trailer on each Toolbox call, the application being profiled will run slower, but not uniformly so. In particular, routines that get called more frequently (for example, time-based routines and event handlers) will appear (in PowerTracer's output) to take more time than they actually do.
- One developer noted significant improvement in his graphics application when he recoded it to use plenty of temporary variables. This is believed to help execution on a PowerPC processor because such code makes use of the PowerPC processor's many registers.

Conclusion

There are no arcane secrets to optimizing your application—but goal setting, profiling, careful analysis, and common sense will take you quite a distance!

One rule to keep in mind is: "Understand everything!" If you don't understand why a certain routine is taking as much time as it is, you may be overlooking a good place to optimize your code. Sure, it's tedious figuring out exactly what a routine is doing, but it's thoroughness that delivers results—and you never know where your next performance improvement is going to come from.

Dig in, and happy optimizing! ♣

Business

Special Marketing Report

Developer Opportunities in the Japan Market

By Kris Newby

It's somewhat surprising that Japan, a country that's roughly the size of California, is the second largest computer market in the world. For Mac OS software and hardware developers, the Japan market is important for more than just its size—it's important because of its affinity for the Macintosh. Today the Macintosh is the second most popular computer in Japan, and Japan is Apple's largest market outside the United States.

If you're already selling Mac OS–compatible products in Japan, congratulations on your excellent timing—chances are your product sales have grown along with the Macintosh line's four-year, 13-point market share climb. (See the graph of the Macintosh market share history on this page.) If you're considering entering the Japan market for the first time, it's not too late. The Macintosh computer's reputation for ease of use, quality, and youth appeal is making it attractive to the new wave of Japanese computer buyers entering this market. In this article I'll discuss the state of the Japan Macintosh market today and offer advice on developer opportunities in the coming year.

The Market Dynamics in Japan

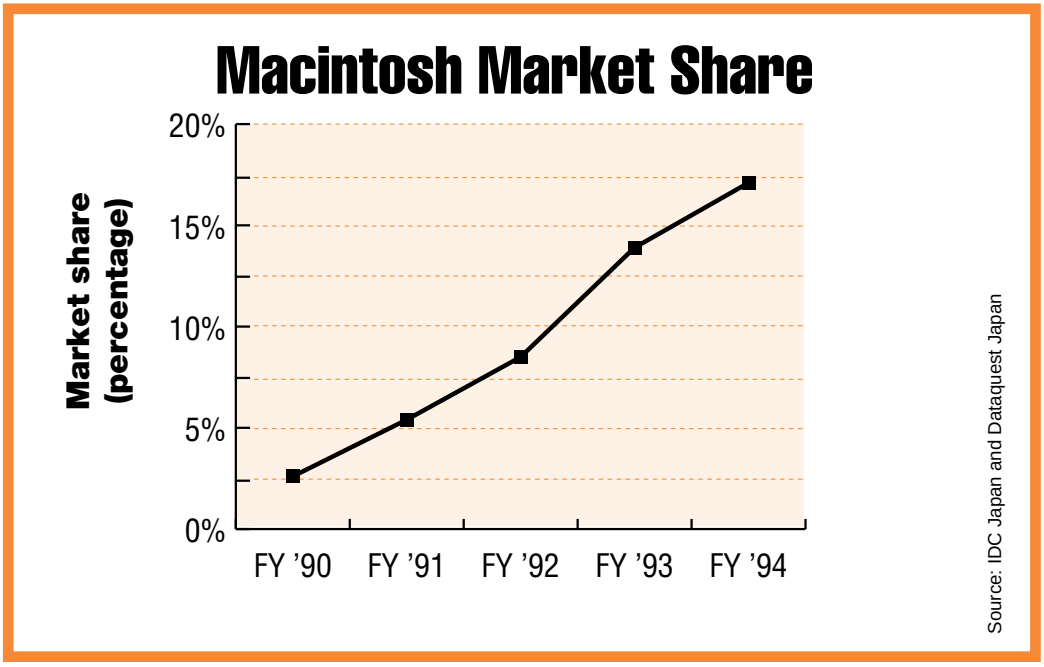
Three years ago, the Japanese computer market consisted of an incompatible mix of high-priced personal computers, medium-priced dedicated word processors, and inexpensive game machines. NEC, using a proprietary version of DOS, overwhelmingly dominated the Japan market, controlling more than 50 percent of market share.

The balance of power shifted, however, with two events that occurred in the early 1990s: IBM Corporation introduced the DOS/V operating system and Apple Computer, Inc., released KanjiTalk System 7 with WorldScript in Japan. The introduction of the DOS/V version of MS-DOS made it possible for standard IBM PC-AT hardware to run two-byte Japanese language software. This also enabled Compaq and Dell to introduce IBM-compatible computers in Japan for roughly half the price of NEC computers, catalyzing a free fall in personal computer prices. On the Apple side of the personal computer market, WorldScript application programming interfaces (APIs) made it easier for foreign developers to localize and sell software in Japan. And together, these events helped shape the following dynamics in today's Japan market:

- *Breakneck personal computer sales.* Falling personal computer prices, catalyzed by what many industry watchers call "Compaq shock," has enabled many more Japanese companies and homes to purchase both IBM-compatible and Macintosh computers, thus increasing the demand for software that runs on these new computers. Projected personal computer sales in Japan are enough to make any software marketing manager smile: In 1994 personal computer sales in Japan grew 35 percent, and sales are expected to grow another 50 percent by the end of 1995, topping the 5 million mark (source: International Data Corporation, or IDC).
- *Cost cutting due to recession.* Personal computer sales are also on the rise as many recession-weary Japanese companies cut costs by replacing obsolete

Inside This Section

Confessions of a New Macintosh Developer 27



mainframes with the new generation of high-powered personal computers. This trend has helped fuel Macintosh and IBM-compatible application sales, as Japanese companies load these new computers with business productivity applications.

• *Growing acceptance of non-Japanese computer products.* Whether it's a result of the strong yen/weak dollar, or the fact that there are now many excellent foreign computer-related products being localized, Japanese computer owners are buying more foreign computer products than ever. A case in point: In 1994 the market share of United States computer companies doubled in Japan, reaching 30 percent (source: IDC).

In today's Japan market, NEC's market share is eroding, and no market leader has emerged in the crowded IBM-compatible side of the business. Apple Japan is the only NEC competitor that has steadily gained market share over the last four years, posting a 16.4 percent market share in 1994, according to IDC Japan and Dataquest Japan (see the graphs on page 23 and 25).

The Macintosh Marketing Advantage

Apple Japan's investments in building a quality brand image, expanding its distribution network, and bringing more third-party software applications to market have also contributed to Apple's healthy growth in this region. Apple Japan has further gained the respect of the Japanese business community by garnering nearly 20 percent market share, greatly increasing the visibility and brand recognition of the Macintosh, and significantly expanding the penetration of Macintosh sales outlets throughout the country.

One competitive advantage that Mac OS developers have in Japan is that the Apple brand has become somewhat of a status symbol.

"In terms of offering quality merchandise, the Apple name is associated with a great deal of status in Japan," says Hikaru Sasahara, president of the Los Angeles–based Interactive Media Agency, a company that acts as a liaison between developers, publishers, and content holders in Japan and the United States. "In the minds of Japanese consumers, the Apple brand is on par in quality and price with Nike."

Similar to Nike athletic gear in the United States, the Macintosh brand has become a symbol of a young, progressive, urban lifestyle. For example, Japanese in their 20s often place Macintosh computers in a visible location in their rooms as an interior design statement. And many television dramas and commercials use the Macintosh as a prop that quickly communicates that the scene is taking place in an urban, "upwardly mobile" setting.

Given the current trends in the Japan market, the reputation of the Macintosh brand is helping the sales of third-party compatible software and hardware in these ways:

• *Ease of use.* With a growing number of first-time buyers entering the market, the Macintosh line's reputation for ease of use and friendliness is helping the sales of entry-level Macintosh computers and software.

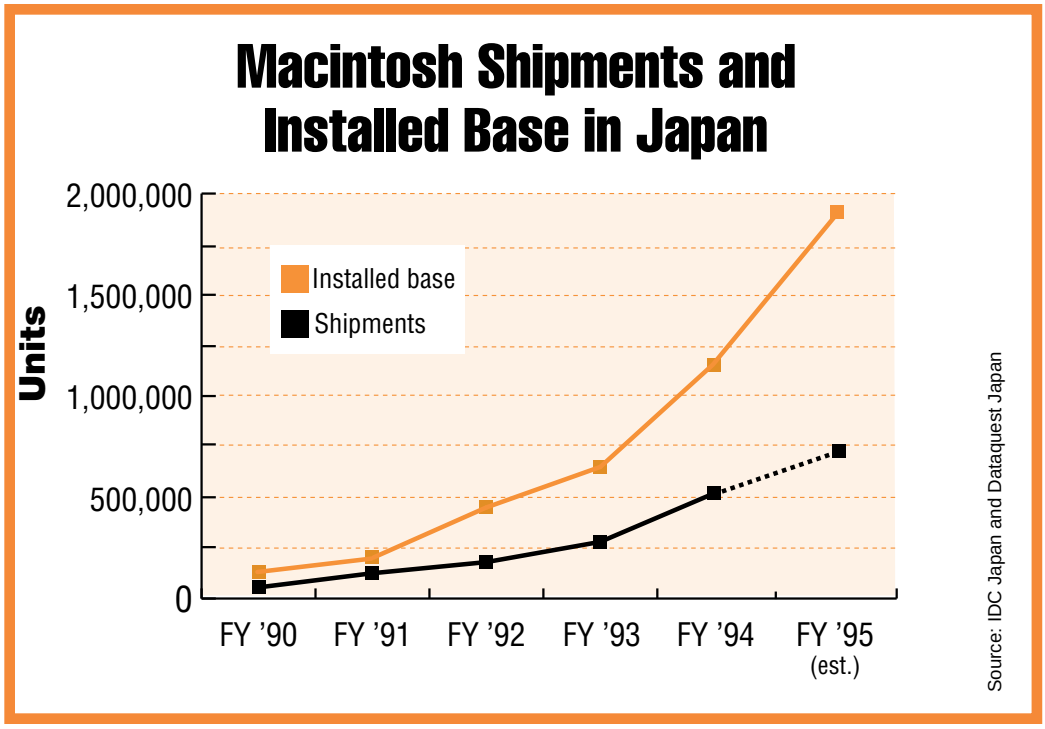
• *Quality.* Developers marketing Macintosh-compatible products benefit from Apple's long track record for quality. Apple maintains this quality level with the help of its "Japan Operations Center" near Narita airport, where every Macintosh is submitted to a grueling set of quality tests before it's released to Japanese distributors and resellers.

• *Youth appeal.* Apple Japan continues to build the Macintosh platform's youth appeal through regular sponsorship of a popular modern music program on Tokyo FM each evening, as well as advertisements in youth-oriented publications and product placements in popular TV shows and movies. Because of the Macintosh computer's popularity with young people, the Macintosh platform provides a distinct advantage to companies marketing games or multimedia content. And it's a good bet that Apple products will be able to hold onto this mind share when this generation reaches a position of power in the coming decade.

Apple takes an active role in helping foreign developers enter the Japan market, with the assistance of a full-time developer marketing organization in Apple Japan. (For more information, see the "Japan Market Resources" box on page 26.) This organization often helps developers find Japanese partners through its business network and Apple-sponsored events. And Apple Japan continues to expand its distribution network, which currently includes 36 authorized distributors and dealers, 18 Apple Centers, and 1,500 authorized retailers (called *MacMasters*).

1996 Japan Market Needs

Apple Japan is focusing its market development efforts on three market areas for 1996: business, education, and consumer. For each area, Apple Japan's market development group is currently gathering customer requirements and defining appropriate solution



bundles of third-party products. Armed with this knowledge, you'll be more able to align your marketing efforts with Apple Japan's advertising and public relations efforts, and perhaps even participate in one of its bundled promotions. Detailed summaries of predicted needs in each market category follow:

• *Business market.* In the business market, there's a need for easy-to-use productivity tools that appeal to the growing number of new Macintosh users. Businesses also need network solutions that tie together installed systems and that help businesses tap into new business opportunities through the Internet. Specific business applications that Apple Japan is looking for include

— simple, easy-to-use business productivity tools such as next-generation spreadsheet, presentation, and data management applications
— collaboration software and PCI-based high-speed networking peripherals
— mobile software and PCMCIA cards for PowerBook computers
— PowerPC "native" scientific, CAD/CAM, and engineering software for Power Macintosh computers
— Internet access software
— accounting, payroll, and inventory control software for businesses
— powerful computer graphics and video editing software for the rapidly growing Japanese multimedia industry

To make it easier for Mac OS developers to sell products into businesses that use Sun Microsystems and Hewlett-Packard UNIX systems, Apple is releasing Macintosh Application Environment (MAE) 2.0 with the Japanese Language Kit. This solution enables users to run Japanese applications in an English-language version of the Mac OS on top of the X Window System™. The Nihongo MAE 2.0 version, due out in the first half of 1996, will let users run Japanese applications in a Japanese-language version of the Mac OS on top of the X Window System.

• *Education market.* The Japanese invest heavily in the education of their children, and with more Macintosh computers making their way into schools and homes, the outlook for educational and "edutainment" software is very promising. The Japanese government is currently trying to introduce computers into many K–12 classrooms, and Japanese consumers are looking for software that makes it easy for students to use computers at an early age.

Sasahara of Interactive Media Agency agrees with Apple Japan's assessment: "The children's market will continue to grow as the shift from video games to personal computers continues. Edutainment, educational, and entertainment software will be in high demand in the future."

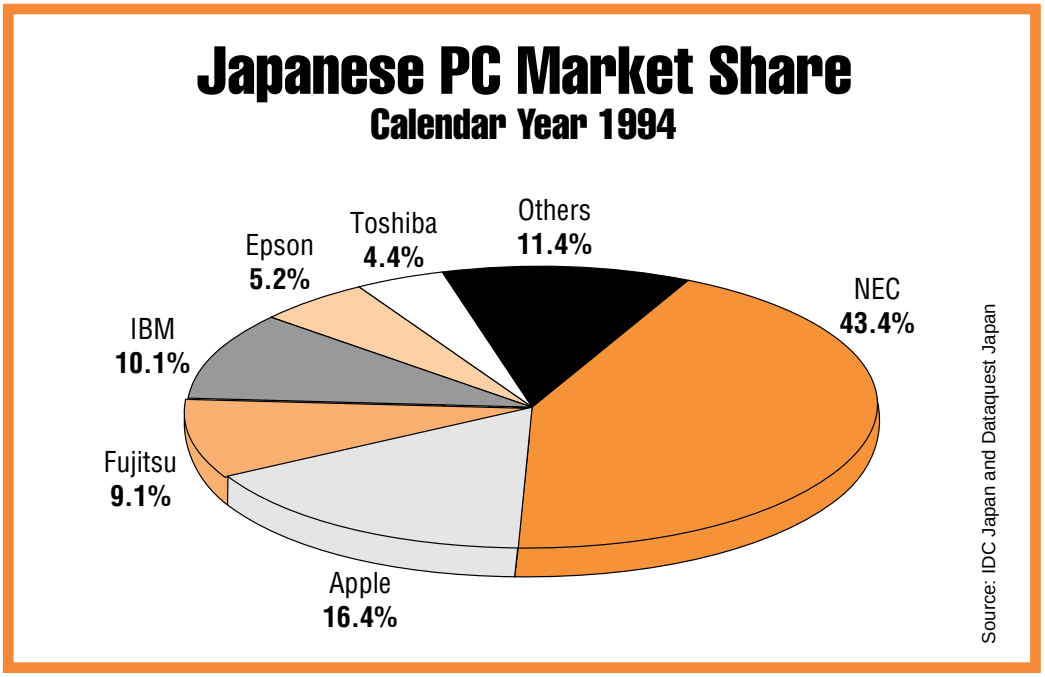
Specific applications that Apple Japan finds promising include

— easy-to-use multimedia authoring tools for K–12 grades
— high-quality edutainment CD-ROM titles
— K–12 education applications
— network software that helps students communicate with others around the globe

• *Consumer market.* The Japanese have always been early adopters of consumer electronics, and with the dropping prices of Mac OS–compatible multimedia applications such as photo, video, and music editing solutions, Apple Japan thinks there's an opportunity for software developers who can create easy-to-use tools for multimedia enthusiasts. For example, Tomoaki Takeuchi, Apple Japan's manager of developer marketing, reports that there are a large number of Japanese users buying high-end Power Macintosh 8100 computers and photo-manipulation software such as Adobe Photoshop, merely for entertainment purposes. Other areas in which Apple Japan has seen increased customer interest include

— communication solutions for Internet access
— solutions that enable Macintosh computers to be used as controllers for audio-visual and synthesizer hardware
— quality CD-ROM game and edutainment titles
— software for editing home videos and music compositions
— home accounting and data management software
— productivity software for mobile workers

In addition to these marketing categories, Apple Japan is extremely interested in helping developers get visibility for OpenDoc-based software solutions. The position of the OpenDoc component software architecture in Japan was further strengthened by the recently announced alliance between Component Integration Laboratories (CI Labs) and IntelligentPad Consortium (IPC) of Sapporo, Japan. (CI Labs is the OpenDoc standards group, and IPC is an association of more than 50 Japanese companies organized to create a worldwide component software standard.) By exchanging memberships, these organizations hope to advance the compatibility of their respective component software



technologies. The bottom line is that this alliance may provide you with an additional way to get exposure for the OpenDoc applications you're working on.

A Hidden Market—Japanese Abroad

Another group of potential customers that many developers don't target is the large number of Japanese-speaking Macintosh users living outside Japan. (It may come as a surprise to you that the country with the largest population of Japanese-speaking people outside of Japan is Brazil.) This market includes everyone from Japanese employees working in foreign countries, to translators, to colleges that teach the Japanese language.

The Macintosh is currently the best multilingual computer system on the market. Simply by loading Apple's Japanese Language Kit onto any Macintosh computer, users can write and edit using Japanese characters without having to install the Japanese version of the Mac OS on their systems. No other personal computer platform can accommodate multiple languages as easily, giving users the ability to input, display, and print text in multiple languages on a single system.

From a development standpoint, Apple's language kits also make it much easier for to you to localize your products for multiple languages. You can hire localizers and test products without having to set up special language-specific systems. And the kits increase your chances of finding local Japanese-speaking users to beta-test your products.

And finally, it's worth noting that once you localize your software for a single two-byte, vertical-scrolling language such as Japanese, the amount of work you'll have to do to localize your software for other growing international markets that use two-byte systems—for example, China or Korea—is minimal.

Pippin Platform Leverage

Pippin, Apple's low-cost CD-ROM playback device based on Power Macintosh hardware and software, provides developers of CD-ROM titles with yet another opportunity. Over the next several years, Apple and Bandai Co. Ltd., as well as several other potential licensees, will work to create a substantial market for Pippin devices—first in Japan, then in other regions.

Richard Sprague, Pippin marketing manager, talks about the potential of this platform: "While a successful Mac- or PC-compatible title may sell hundreds of thousands of copies, in the low-end game world of Nintendo or Sega many titles sell *10* or *20* times more than this. We think Pippin is a great opportunity for any developer who's really interested in this low-end consumer market—that is, the 50 percent of U.S. households and the 60 percent of Japanese households who have no interest in buying a traditional personal computer."

Getting Macintosh CD-ROM content to run on the Pippin platform, in most cases, requires very few code changes, so for this category of developer, the Pippin becomes yet another channel for leveraging investments in existing Mac OS titles.

Bandai will begin test-marketing its Pippin device in December 1995, and then will make it more widely available in March 1996, according to Bandai sources. Later in 1996, the company will begin selling it into other markets. Bandai has publicly stated that it expects to ship 500,000 devices in the first year. (See the "Japan Market Resources" box for more information on Pippin.)

Getting Started in Japan

Foreign software and hardware companies that have met with success in Japan almost universally agree that finding a Japanese partner is an essential step to getting started. By teaming up with a Japanese republisher or distributor who really understands the Japan market, you'll be able to more effectively establish a strong pricing and distribution strategy. And more important, you'll have a local partner to help with everything from creating ads and brochures, to establishing strong relationships with the local Macintosh press, to running product training classes.

GraphSoft took this approach when introducing MiniCad, a 3D computer-aided drawing application, to Japan. In 1989 a Tokyo-based developer/distributor, A&A Co., Ltd, asked GraphSoft if they could localize MiniCad for the Japan market. Today MiniCad is the best-selling CAD program in Japan.

Richard Diehl, president of GraphSoft, offers advice to other developers entering this market: "The trick to succeeding in Japan is finding a strong Japanese partner. Our partner A&A not only gave MiniCad the market presence that it needed in Japan, but they did an excellent job of localizing our product for the Japanese language and culture."

Japan Market Resources

Articles and Documents

- *A Guide to Japan for Macintosh Developers 1995* includes helpful information on entering the Japan market, as well as an extensive list of Japanese distributors, republishers, authorized resellers, Macintosh publications, events, and more. It's available on the *1995 WWDC Presentations* CD in the Apple Pacific folder.
- "The Emerging Asian Software Market—Investing Today for Big Returns Tomorrow," *Apple Directions,* September 1994, page 25.
- "Pippin: A New Platform for Multimedia Titles," *Apple Directions,* May 1995, page 17.

Other Resources

- *Apple Japan Contact.* For more information about the Japan market, send a message to Tomoaki Takeuchi, developer marketing manager, at takeuchi2@applelink.apple.com or at the fax number 81-03-5411-8579.
- *Localization for Japan.* This document provides you with technical guidelines for localizing products for Japan. You can download it from eWorld using the path Computer Center:Developer Corner:Apple Developer Services:Reference Library:Programs & Marketing:Localization for Japan.
- *Pippin Web page.* For the latest information on the Pippin platform, check out the Pippin page on the World Wide Web at the location http://pippin.apple.com.
- *Interactive Media Agency.* This Los Angeles–based company acts as a liaison between developers, publishers, and content holders in Japan and the United States. Its president, Hikaru Sasahara, can be reached at 310-477-9923.

As was the case with A&A, there are a number of Japanese distributors and resellers looking to localize innovative foreign products for Japan. Though many of the large distributors, such as Softbank, Software Japan, Computer Wave, and Seiwa Systems, don't like to deal directly with foreign companies, there are many Apple Japan Authorized Distributors with whom you can work to deliver products to the Macintosh channel.

Apple Japan's developer group can help you research and develop relationships with these partners. As a first step, Apple Japan suggests that you read the document *A Guide to Japan for Macintosh Developers.* (See the "Japan Market Resources" box on page 26 to find out how you can get a copy.) This document includes helpful information on entering the Japan market, as well as an extensive list of Japanese distributors, republishers, authorized resellers, Macintosh publications, and more. Then, Apple Japan recommends that you send a letter of introduction to the Apple Japan authorized distributors listed in this document, explaining that you're interested in discussing product distribution. Also include a description of your product, a product demo, and company background information in the letter. Feel free to query Apple Japan if you need specific information about any of the listed companies.

Before you contact Japanese partners, it helps to understand Japan's retail channel rate structure. Though margins and distributor discounts vary widely depending on the type of product you're selling, negotiations usually start at this point: Distributors and republishers typically purchase your product for 50 percent off the suggested retail price (SRP), taking a 10–15 percent cut of the SRP before passing it on to the dealer/reseller. The dealer/reseller, in turn, may sell the customer a product at 20–25 percent off the SRP. Many companies give the distributor an additional ten market points for handling localization, although most find that it's better to negotiate a fixed localization fee with a specialist, rather than cutting into their royalty stream.

You also may want to get a feel for advertising rates. One of the popular Macintosh magazines in Japan, *Mac Fan,* states that a standard one-page color ad placed in a single issue costs 500,000 yen (about U.S. $4870), and a back cover page ad, which is typically the most visible and expensive spot in a magazine, costs 1,200,000 yen (about U.S. $11,680).

Small Country, Big Opportunity

Mac OS developers looking to expand into overseas markets should seriously evaluate the Japan market as a first step. Macintosh sales in this country are growing at a rapid rate, and the Macintosh brand is perfectly positioned to appeal to the demographics of the new wave of users. Developers entering this market also benefit from Apple's helpful developer organization and Apple's reputation for delivering quality products. ♣

Kris Newby (newby2@aol.com) is a technical communications consultant and freelance writer based in Palo Alto, California. A wealth of information was also provided by Tomoaki Takeuchi (takeuchi2@applelink.apple.com), Pat Kirkish from Apple Pacific, and Richard Sprague (sprague1@applelink.apple.com), Pippin marketing manager.

Confessions of a New Macintosh Developer

By Paul Baldwin, Domark Software

I was a Macworld virgin. But I wasn't worried. After all, I'd been working in the sales and marketing department at Domark Software for more than seven months. And in the fast-paced computer game industry, where experience is accrued in dog years, I was practically a veteran.

My mission at Boston Macworld 1994 was simple—demo Domark's first Macintosh game title, *Flying Nightmares,* in Apple Computer's snazzy little Power Macintosh pavilion, and tempt the Mac faithful with our forthcoming release, *Out of the Sun.*

I must say, I was feeling fairly confident. Domark, a London-based company, had been in the gaming business for nine years before we created our first Macintosh title. We'd experienced success on the gamut of game platforms: Amiga, Atari 2600, Commodore, DOS, Sega, and Nintendo. And though Domark had only recently entered the U.S. market, our reputation for high-end, accurate flight simulators was growing rapidly.

As the first "native" entertainment title shipping on the Power Macintosh computer, I'd been wined and dined by the Power Macintosh launch team in New York City. Units were flying off the shelf. I was ready to take Beantown by storm.

At last, the Macworld public flowed into the Power Macintosh pavilion. I prepared myself to humbly accept praise. Instead, I was met with a totally unexpected set of reactions:

"*Flying Nightmares* is a $@#%*& DOS port!"

"Your interface stinks!"

And my personal favorite: "I love your game, but your $@#%*& fonts are *so* PC."

Needless to say, I really didn't expect such a *warm* response. Though my story may be a bit exaggerated (after all, I *am* in marketing), the message I needed to bring back to Domark was loud and clear—the Mac OS market is an entirely different beast than other markets.

Lesson #1: No DOS Shovelware

Even though the Mac OS market was starving for games at the time of our product launch, users let us know that they weren't going to be force-fed just anything. They liked our game, but we had missed something in our Macintosh implementation.

First, I learned that straight "ports" of games from other platforms are taboo. For success in this market, we needed to take advantage of the technologies built into the Macintosh. We needed sharp, intuitive Macintosh-style interfaces. And we had to clean up our "jaggy" DOS fonts and graphics.

Second, I began to appreciate how grass-roots and fanatical the Mac OS market is. If you make a good game, and you win over the hard-core Macintosh enthusiasts, half your marketing battle is won. Despite the *unusual* way some

Domark's upcoming title, *Absolute Zero,* is a science fiction game in which mining colonists and aliens battle for control of Europa, one of Jupiter's moons. The interface for this multiplatform game was created "from the ground up" on the Macintosh.

Macintosh enthusiasts demonstrated their affection, the show was an incredible source of marketing information. Gamers, resellers, user group leaders all went out of their way to offer advice. And it all revolved around one central theme: "We don't want a lot of crap. We want solid, well-thought-out titles."

Lesson #2: Lower Marketing Costs

Even though *Nightmares* was what Macintosh users called a "DOS port," our sales were remarkable. In the first six months *Nightmares* sold over 35,000 copies. It was such a great success that we immediately began porting over our World War II flight simulator, *Out of the Sun,* and it won the *Inside Macintosh Games* "Flight Simulator of the Year" award in 1994. All this, and we were only advertising in the Macintosh mail-order catalogs and *Mac Home Journal.* Soon we realized that we'd sold all these copies for a quarter to half of what we'd spent marketing in the DOS market. We found that, compared to the other platforms we had been developing for, the Mac OS market provided us with these advantages:

- *A low barrier to entry.* Our initial development and marketing budget for *Flying Nightmares* was just under $50,000.
- *An open market.* Despite the rapid flow of new titles in the Mac OS game market, it's still wide open compared to the crowded DOS and cartridge game markets. *Nightmares,* for example, was the first new Macintosh flight simulator on the market for almost two years at the time of its release. Currently game categories such as sports and role playing are still untapped on this platform.
- *A favorable distribution channel.* Macintosh catalogs, such as *MacWarehouse, MacZone, MacConnection, Mac's Place,* and *MacMall,* are an excellent, cost-effective way for new Macintosh developers to distribute products. Today 40 to 50 percent of our sales come from catalogs. We've also enjoyed a great affiliate label relationship with Spectrum Holobyte, a major game developer and distributor. As a result of this relationship, we've been able to sell *Flying Nightmares* into most of their retail accounts. To date, our Mac OS products are on the shelves of nearly all major U.S. computer chain stores. And having a Mac OS line helped us get into Sears and Office Depot, a feat that we weren't able to fully accomplish with our PC line.
- *A longer product shelf life.* Because the PC-compatible side of the market is so competitive, PC products are typically removed from the reorder list of major resellers after about six months to make room for new titles. Our Mac OS version of *Flying Nightmares,* on the other hand, continues to sell well after almost two years—because it's a great game and the Macintosh Performa is selling so well. This holiday season we think *Nightmares* sales will top the 50,000 mark.
- *Lower support costs.* I know "plug-and-play" is a bit overused these days, but the Macintosh computer's plug-and-play capabilities really mean you'll spend less on support. Our technical support people love the Macintosh because there are no Sound Blaster cards or incompatibility problems to worry about. On average, for every three PC users that call in for support, we only get one Macintosh caller.

Lesson #3: An Accessible User Community

A key factor in Domark's success was our relationship with Apple's Power Macintosh evangelists. Not only did Apple provide our technical team with assistance, including a few onsite visits to Domark, but they offered some much-needed marketing insights.

Our first big break was an invitation to the Apple Power Macintosh press tour in New York City. We were one of 30 developers (and the only game developer) invited to demo new native Power Macintosh applications to the world's computer press. Needless to say, I soon had all the journalists fighting in Harrier jets. It was during this event that I came to understand fully the community aspects of the Mac OS market. Unlike the abyss of the PC-compatible market, it appeared that a small, agile developer could more easily comprehend and attack the Mac OS market. And with Evangelism's help, we could quickly establish ties with Macintosh user groups, the retail channel, and ultimately core Macintosh customers.

Macintosh user groups have also been very influential in letting the Macintosh community know about our product. Accessible at group meetings, at conventions, and online, user group members are both your toughest critics and best friends. Domark now works closely with these groups when marketing our titles. We spread the word about *Flying Nightmares* through the User Group Connection (a Scotts Valley, California, organization that offers direct-mail services to user groups), phone calls to group members, and announcements at Macworld user group meetings. Many of these groups have newsletter editors who love to write reviews on new products. We have distributed literature, demos, and finished products to these organizations, all with great results.

Lesson #4: A Mac-First Strategy

With a year of Mac sales experience under our belts, we brainstormed about what to do next. Colin Boswell, Domark's lead programmer, and John Kavanagh, our vice president, came up with a great idea for a science fiction game. At first the plan was to slap together a quick-and-dirty DOS game around the concept, reusing some of our custom 3D graphics technology. But once we got into the game planning, we

realized we weren't doing the idea justice. Very soon the project went from a quick port to the intricate game that we're about to ship, called *Absolute Zero.*

Domark has always taken a multiple-platform approach to product development: We start with our library of game primitives, develop a common code base in C, then add platform-specific parts of the game at the end of the process. Whereas in the past we've created games from a very DOS-centered point of view, our Macworld experience made us think twice about this approach. We knew that if we could earn a great return off our Mac OS products with just a "DOS port," sales would be even better if we created an Evangelism-blessed Mac OS game. So we added a Macintosh programmer, Mike Kelly, to our development team to make sure we were getting our Mac OS product right.

Before long we learned that the Power Macintosh made it easier to soup up our interface and go all out with our story line. We found we couldn't live without parts of the Macintosh Toolbox, so Mike recreated them on the DOS side. And with 100 percent of the game's interface designed on the Macintosh, we decided that it made more sense to release our product on the Mac OS first.

Now, as I go on sales calls with *Absolute Zero,* I realize that by breaking with tradition and introducing our game on the Mac OS first, we're going to really get the attention of the Macintosh game community. Ultimately the reviews and subsequent word-of-mouth that we generate from the Mac OS market will help us when we introduce a Windows version this fall.

Lesson #5: Just Do It

My advice to developers considering a move to the Mac OS platform is *just do it.* Compared to the PC market, introduction costs are lower, the market is less crowded, and the products have a longer shelf life. Domark has found that by deploying products to the growing Mac OS market, we essentially cut our per-unit development cost in half. In fact, we've been so happy with the sales of our Mac OS–based games, that we're introducing three more games to this market in the coming year.

Domark's new Mac OS game titles are now created from the ground up by our Macintosh development team. As Macworld San Francisco approaches, I'm all the wiser, though I must admit that I'm still anxious to see how the Macintosh faithful display their affections. ♣

Paul Baldwin (paulbald@domark.com) is Macintosh marketing manager at Domark Software, based in San Mateo, California.

A Game Programmer's Perspective on the Mac

Colin Boswell, Domark's development manager and lead programmer, is a multiplatform programmer who credits his "religious" conversion to the Mac OS platform to a misunderstanding.

"I was bored one weekend, so I started messing around, trying to get a rotating airplane demo to run on a Mac," says Boswell. "Within a couple hours I got it working, so just for the fun of it, I showed it to our vice president John Kavanagh. Next thing I know, I get a note from our president, Dominic Wheatley, asking when our Mac flight simulator is going to be ready. Then I find out that John is already talking to Apple evangelists about creating a hot new game title for the Power Macintosh product launch. So four months and many drinks later, much to the amazement of us all, our product was ready for Apple's launch.

"The most important thing we learned coming over to the Macintosh platform was to get early feedback from people who understand the Macintosh user interface," says Boswell. "Macintosh users loved playing our first game, *Flying Nightmares,* but we received a lot of criticism on its DOS core and low-resolution graphics."

The interface of Domark's upcoming game, *Absolute Zero,* was written entirely on the Macintosh. Domark's Macintosh expert, Mike Kelly (you may have played his shareware games Cyclone and Space Madness) talks about developing simultaneously for the DOS and Mac OS platforms: "There were actually a couple of pieces of the Macintosh Toolbox that I couldn't live without, such as the Resource Manager, so I duplicated them for use on the PC."

"The more I use the Mac," says Boswell, "the more I like it, which is quite perverse, since the low-level code I write is entirely platform-independent. Best of all, the Mac platform tools are great—I wish the DOS side had something like Metrowerks Code Warrior. And Debabilizer is God's own tool."

Boswell adds, "Game designers are all frame-rate junkies, and I think the slow speed of the Mac 680x0 processors originally disillusioned many DOS game developers. But with the speed of the Power Macintosh and the astounding Macintosh sales we've seen, DOS game developers would be foolish not to take another look at the Macintosh platform."

Listings

Developer University Schedule

Apple Computer's Developer University courses provide "how-to" instruction in all aspects of programming the Macintosh computer. With course offerings on a broad range of subjects, from PowerPC and OpenDoc to Newton and MacApp, Developer University (DU) will give your engineers information they need to build Macintosh and Newton software using the latest Apple technologies.

The following is the schedule of upcoming DU course offerings. Courses marked with an asterisk (*) are held in Portsmouth, New Hampshire. All other classes are held at the Apple R&D Campus in Cupertino, California.

October

Newton Programming: Essentials	10/23–10/27	$1500
Programming QuickDraw 3D	10/23–10/25	$1000
QuickStart Mac OS Programming	10/23–10/27	$1500
Macintosh Debugging Strategies & Techniques	10/30–11/1	$900
Scripting with AppleScript	10/30–10/31	$600

November

Creating Apple Guide Help Systems	11/6–11/9	$1200
Multimedia Development with QuickTime VR	11/7–11/9	$1500
PowerPC BootCamp	11/6–11/9	$1200
Apple Events/AppleScript Programming	11/13–11/17	$1500
Programming OpenDoc	11/13–11/15	$1200
Programming with OpenDoc Development Framework	11/13–11/17	$1600
Advanced C++	11/27–12/1	$1200

December

Programming with QuickDraw 3D	12/4–12/6	$1000
QuickStart Mac OS Programming	12/4–12/8	$1500
Multimedia Development with QuickTime VR	12/12–12/14	$1500
Newton Programming: Essentials	12/11–12/15	$1500
Programming OpenDoc	12/11–12/13	$1200
Scripting with AppleScript	12/11–12/12	$600

To register for a class or to get a complete course description by fax, call the Developer University Registrar at 408-974-4897.

Course descriptions can also be found electronically at the following locations:

AppleLink: Developer Support:Developer Services:Apple Information Resources:Developer Training: Developer University

eWorld:Computer Center:Apple Customer Center:Apple Developer Services:Developer Information: Developer University

Internet: http://www.info.apple.com/dev

America Online: Computing:Computing Forums:Development: Mac Development Q&A:Developer University ♣

The Internet Page

This feature is devoted to informing you about where you can go on the Internet for online information about Apple Computer, Inc.; its products, technologies, and programs; Mac OS and Newton programming; and other subjects that pertain to the business of computer product development.

You'll find this feature particularly helpful when you view it at the Apple Directions *Web page (located at http://www.info.apple.com/dev/). There, all the names of the locations listed in this article are linked to the sites themselves; clicking the names will take you directly to the relevant Internet location. We'll update this feature every month, based both on what Apple is doing on the Internet and on your feedback.*

Apple Sites

This section describes World Wide Web sites maintained by Apple Computer.

http://www.info.apple.com/dev/

This site contains the Apple Developer Services and Products page, and is probably the most important World Wide Web page for you. Not only does it contain the online version of *Apple Directions,* and *develop,* the Apple Technical Journal, but this page also links you to a variety of other sites that give you access to the gamut of Apple's online developer support services.

http://www.apple.com/

This site contains the Apple Computer home page, with links that will eventually let you get to just about all the other Internet sites maintained by Apple, even the ones listed separately here.

http://www.austin.apple.com/macos/

This is the Mac OS Web site. You can go here for the latest information on the Mac OS, including details about Copland, white papers on new Mac OS technologies, marketing and strategic information, and other items to help you develop new Mac OS products.

http://www.apple.com/whymac/

The official source for official Apple ammunition to fight the war against Windows 95, including the extensive series of Windows 95 vs. Macintosh Updates, prepared in the wake of the Windows 95 release.

http://www.info.apple.com/gomobile/
This site contains complete information about PowerBook computers and the full line of Apple mobile computing solutions.

http://www.info.apple.com/opendoc/
This is the site of Apple's OpenDoc home page, featuring Developer Depot, where you can find the latest OpenDoc release, documentation, and tools, and Developer Showcase, from which you can download and sample actual OpenDoc parts!

http://www.info.apple.com/dev/thirdparty/
Apple Fellow Guy Kawasaki set up this Web page to list your hardware and software products. Fill out the form located at the site to add your products; that way, everybody on the 'net can find out about what you're up to.

http://www.info.apple.com/dev/evangelism/powertalk/
Apple's PowerTalk home page, with resources for PowerTalk programmers.

http://www.info.apple.com/qd3d/
Apple's QuickDraw 3D home page contains everything you need to know about QuickDraw 3D, including QuickDraw 3D applications you can "test drive."

http://www.info.apple.com/powermac/powermac.html
http://www.info.apple.com/ppc/ppchome.html
These are two useful sites for information about Power Macintosh computers.

http://quicktime.apple.com
This site contains the QuickTime Continuum page with news and technical and marketing information about QuickTime.
http://qtvr.quicktime.apple.com
This is the location of the QuickTime VR page.

http://www.info.apple.com/gx/gx.html
This site contains the QuickDraw GX home page.

http://www.apple.com/documents/otherappleservers.html
This is the site of the Apple Internet Servers page. Once you've exhausted the obvious Web sites just listed, this page will give you ideas about where else to go on the Internet to find the information you need. This page includes lists of other Web sites as well as Gopher and FTP sites.

Non-Apple Sites

We can't guarantee the information the following sites contain, since they're not created by Apple, but we think you'll find them useful and interesting—even entertaining.

http://www.guideworks.com/
This non-Apple site is the location of the guideWorks home page; it contains so much information about Apple Guide that you can think of it as the Apple Guide home-away-from-home page.
http://www.astro.nwu.edu/lentz/mac/programming/tools.html
This non-Apple site is a terrific source for Apple and non-Apple Macintosh programming tools.

http://west.ucd.ie/
The Web site for University College Dublin—WEST (Web Educational Support Tools), winner of the recently announced Apple Enterprise Awards in the Client/Server—Education and Government category. Information about their work—and demos of their software—are contained at this site.

http://home.mcom.com/home/internet-search.html
This site contains the Internet Search page, which gives you access to InfoSeek, Lycos, and WebCrawler, three excellent Web search engines. If you use Netscape, you can reach this location just by clicking the Net Search button.

New This Month/From Our Readers

These are locations that we've just become aware of, thanks to *Apple Directions* readers inside and outside Apple. Know of a particularly useful site? Whether it's a Web page, an FTP site, or a newsgroup, let us know about it and we'll consider adding it to this feature next month, along with your name (!). Send your suggestions to the Internet address a.directions@applelink.apple.com.

http://www.amp.apple.com
This is the site of the Apple Multimedia Program (AMP) home page. If you're a multimedia developer (or considering getting into multimedia), you'll want to check out the information on this page about Apple's multimedia technologies, as well as the links provided to other Internet sources about multimedia. It also includes the AMP Member Showcase, a searchable database of multimedia developers.

http://www.info.apple.com/education
Here's where you'll find the Apple Education home page with information about Macintosh computers for the education markets. You can also use online forms located at this site to request product specifications, information about Apple Education Series (bundled products), and technical support from Apple engineers.

http://www.mae.apple.com
The Macintosh Application Environment (MAE) home page.

ftp://ftp.sri.ucl.ac.be/pub/
A French reader alerted us to this interesting FTP site, where you can find French versions of Macintosh Internet software, including Eudora, Fetch, Finger, FTPd, Gopher Surfer, NCSA Mosaic, NCSA Telnet, TurboGopher, and many other applications. An affiliated World Wide Web site (http://www.sri.ucl.ac.be/SRI/jpk/logIntMacFr.html) describes—*en Français*—what's available at the FTP site. (Thanks to Jean-Pierre Kuypers for letting us know about this location.)

http://www.kaidan.com
Here's an interesting reader-recommended site that will be especially interesting to QuickTime VR developers. The site contains information about add-on lenses and QuickTime VR camera mounts for QuickTake cameras. (Thanks to Dave Case.)

http://www.class.com/MacTech/URLs.html
This site contains a useful list from *MacTech* magazine of Internet locations on a variety of subjects, most of them pertaining to the technical aspects of Mac OS development.

http://www.freepress.com/myee/ultimate_mac.html
This site contains the ULTIMATE Macintosh page, including more Mac OS information and software than you could possibly imagine exists. We think you'll particularly enjoy the software archives and games sites, from which you can download real-live software and play with it.

http://www.cs.brandeis.edu/~xray/mac.html
Nathan's Everything Macintosh page is another treasure trove of Macintosh information; it contains a thorough listing of Apple and other corporate sites that pertain to Mac OS development as well as games, e-mail mailing lists, periodicals, a listing of FTP sites, software archives, and even Apple II information.

http://www.cilabs.org/
The location of the CI Labs home page, which contains a great deal of OpenDoc content.

http://rever.nmsu.edu/elharo/faq/vendor.html
The Macintosh Vendor Directory, a directory of companies that make and sell products for the Macintosh computer.

http://www.metrowerks.com/
This is Metrowerks's Web site, with information about its Code Warrior PowerPC development tool.

http://www.nisus-soft.com/~nisus/
The location of the Nisus Software home page, which we list partly because of its clever layout. The page looks like a Macintosh desktop; clicking the icons on the desktop takes you to Nisus's various Web postings. Just for fun, click the Trash icon and see where you end up! ♣

Know of a particularly useful site? Whether it's a Web page, an FTP site, or a newsgroup, let us know about it and we'll consider adding it to this feature next month. Send your suggestions to a.directions@apple.applelink.com.

Errata and Clarifications

Here at *Apple Directions* we want to be as accurate and complete as possible, to give you the best possible information on which to make your business decisions. Here are some items that have come to our attention:

• In the September 1995 Strategy Mosaic, "Seeing Through Windows—And Into the Future," in the table "Macintosh Innovations and Wintel 95" on page 6, the item in the second row, "3 1/2-inch floppy disk drive support," was printed in the wrong column. It should be in the column labeled "Technologies present on Wintel 95 and Macintosh platforms."

• In that same Strategy Mosaic, in the "Beginning the Next Ten Years" table on page 8, author Gregg Williams inadvertently left speech input and output (in the form of Apple's PlainTalk technology) out of the table of important future technologies that Apple Computer, Inc., has pioneered. Gregg says he is particularly chagrined by this omission, since the speaker-independent speech-recognition component of PlainTalk is one of his favorite Apple technologies.

• The September 1995 news item "Apple to Offer Oracle Power Objects for Client/Server Development" makes the statement that "Visual Basic . . . contains no built-in support for database access and manipulation." Reader Jerome Villafruela of Echirolles, France, wrote *Apple Directions* to say that "This is false—Visual Basic can access dBase, Paradox, and Access databases. The Pro edition comes with ODBC [Open Data Base Connectivity] support and additional data-aware controls."

Mike Zivkovic, product marketing manager for Apple development tools, confirmed Mr. Villafruela's statement: "The database comment in the Oracle Power Objects (OPO) article in *Apple Directions* needs clarification.

"The statement about Visual Basic containing 'no support for database access and manipulation' was referring to *native* database access (that is, support for Oracle SQL•net and Sybase OpenClient) which is available out-of-the-box in OPO and is not a part of the Visual Basic package. The native database access, unlike ODBC (which is a part of Visual Basic), enables high-performance client/server applications and effective use of database-specific functionality. Customers who are building robust and high-performance applications are always turning to native database connectivity. ODBC is adequate for *ad hoc* queries and less demanding decision-support applications.

"The database manipulation part of the statement needs to be clarified as well. OPO includes very easy-to-use and powerful predefined database objects. These objects enable developers to create programs by dragging and dropping objects instead of having to write hundreds of lines of Visual Basic code. In addition, OPO automatically takes care of the mechanics of connecting client/server applications to the Oracle, Sybase, or MS SQL Server databases. Developers do not have to be involved with a long process of learning how to correctly access, manage, and manipulate data in a client/server environment.

"A good way to learn more is by browsing Oracle's Web page at http://www.oracle.com, downloading a demo, or reading the OPO white paper [available from the Oracle Web site]. Clearly, the best way to learn about the merits of Oracle Power Objects is to use it and compare it against Visual Basic or any other comparable tool." ♣

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 U.S. APDA office from abroad, the number is 716-871-6555. You can also reach us by AppleLink at APDA or by e-mail at APDA@applelink.apple.com. More detailed APDA ordering information is available at the following locations:

- Internet: http://www.info.apple.com/dev/apda.html
- AppleLink: Developer Support:Developer Services:APDA
- eWorld: in the Developer Corner of the Computer Center