Editor’s Note:

Industry Watch: True Stories

Amber Provides Path to Compound Documents, Modular Software

Apple Ventures Into Electronic Publishing

Apple Opens UNIX Market to Macintosh Developers

Apple WorkGroup Server Special Offer

WWDC Tapes Available

QuickTime for Windows 1.1

New PowerBooks

Don Norman

Tech Notes Changes

CD Highlights

Human Interface

Taking Your Applications Into the Open Systems Market

AppleScript

Developer Outlook: The GamesMarketing Game

It Shipped!

Hot Products From Apple

APDA Top Ten

GetNextEvent

High Marks For Macintosh on PowerPC Compatibility

Compatibility is Apple Computer, Inc.’s number-one priority for the upcoming PowerPC-based Macintosh systems. Customers and developers have made significant investments in software, and Apple’s goal is to protect that investment by ensuring that 680x0-based Macintosh software will run unmodified on PowerPC-based Macintosh systems when they become available in the first half of 1994.

Toward this end, Apple plans to be aggressive in testing today’s applications for compatibility throughout the entire development process, starting now, and continuing through introduction. Apple will make this possible by hosting Macintosh on PowerPC compatibility labs at developer events around the world and by making prototype Macintosh on PowerPC systems available for testing in permanent Apple compatibility labs worldwide later this year.

Apple opened the first PowerPC compatibility lab at the 1993 Worldwide Developers Conference, where developer attendees could quickly assess the level of compatibility of their products on early prototype PowerPC Macintosh systems. More than 600 applications were launched and run on the prototype systems, and more than 90 percent of them ran without a hitch, demonstrating the solidity of the emulation technology.

Encouraged by these early results, Apple is committed to ensuring similarly high levels of compatibility as testing continues, and becomes increasingly in-depth, over the coming months.

Compatibility of 680x0-based applications is made possible through a 68040 (no floating-point unit) software emulator that will be an intrinsic element of all PowerPC-based Macintosh systems. Depending on the application, customers can expect their off-the-shelf 680x0-based applications (those that have not been recompiled for PowerPC) to run on their PowerPC-based Macintosh at speeds comparable to a 68040-based Macintosh.

In addition, customers will be able to seamlessly run 680x0-based and native PowerPC-based Macintosh applications side by side on their PowerPC-based Macintosh systems by means of a mixed-mode architecture. (For more information, see “Macintosh on PowerPC: Top Developer Q&As” in the June 1993 issue of *Apple Directions*.)

There are many steps developers can take today to ensure compatibility of their applications with the PowerPC-based Macintosh systems and to specifically take advantage of the new architecture. For a list of these steps, see “Preparing for PowerPC: Ten Commandments,” also in the June 1993 issue of *Apple Directions.*

There will be opportunities to test applications on a prototype PowerPC-based Macintosh system at this summer’s Apple developer conferences in Germany, France, and Japan, at Macworld Boston in August, at Macworld San Francisco in January, and at Apple compatibility labs worldwide in late 1993. Stay tuned to *Apple Directions* for more information on these opportunities.

Amber Provides Path to Compound Documents, Modular Software

At the Worldwide Developers Conference in May, Apple Computer, Inc., gave the first details of Amber, an open, cross-platform, compound-document architecture. Amber technology will be seeded to developers in late 1993, and it is targeted for simultaneous release on the Macintosh and Windows platforms (including Microsoft’s OLE 2.0) in the middle of 1994. Amber provides the framework for something Apple’s been talking about for a while, *modular software*—that is, a collection of small programs, each of which can do one thing well: for example, styled text, spreadsheets, or graphics.

Currently, users create documents in an application-centered way—they use multiple applications to create different parts of the final document. They must then cut and paste most of the parts from different applications into a document governed by one specific application (often with less-than-perfect results).

Document-Centered Computing

Amber lets the user focus on the task to be done—creating a document—rather than juggling the applications needed to create the final document. The paradigm of Amber is that of a document that contains multiple parts, each of which does one thing well. (Amber parts are not related to the entities named *components,* which are governed by the Component Manager.) The visual representation of these parts can overlap on the video display and do not have to be rectangular. Parts can be linked, so that when the user changes one part (a spreadsheet, for example), the part to which it's linked (say, a bar chart) automatically updates itself.

The closest Amber equivalent to the traditional application is called a *part handler,* and there are two kinds: *part editors* (which can display, manipulate, and print the part just clicked) and *part viewers* (which can only display and print the part). Apple strongly encourages you to make your viewer freely available; in this way, any users can view and print any Amber document, even if they don't buy the part editors that you sell.

Amber also provides editing "in place"—that is, the user edits a part where it is within the document. When the user clicks in an Amber document, that part's editor becomes active, changes the items in the menu bar, draws a frame around the content of the part just made active, and provides the interface that the user interacts with.

An Open, Cross-Platform Standard

Amber will be an open technology. Apple will make Amber source code available to all vendors who want to implement its architecture in their products, and will disclose the file structure for Bento, the storage model that Amber will use. With this information, anyone can write part editors and viewers for any platform that supports Amber,
and part editors and viewers on different platforms can read the same Amber file. Amber's open design and its document-centered, cross-platform orientation make it a "vendor-neutral" environment, and Apple is working with other companies to make Amber an industry-wide standard.

Easy to Customize

Amber provides customization on several levels. By its very nature, an Amber document (perhaps in the form of a template, meant to be filled out by a user) can unite different parts—for example, a quarterly report that includes text, a spreadsheet of financial data imported from a remote database, and corresponding graphs—into the seamless solution that the user needs. In this way, both solution providers (value-added resellers, consultants, and others) and individual users can easily create computer solutions.

Amber supports the Open Scripting Architecture (OSA). This gives both solution providers and individual users another way to build powerful, custom solutions. By using AppleScript or other scripting systems, they can orchestrate complex interactions between parts.

You can create your own suite of parts (or license them from other companies) and combine them in different ways to create multiple products, each for a different audience. For example, from your own collection of parts, you might be able to create an entry-level word processor, a multiple function “works” program, a specialized word processor for the legal profession, and a medium-level page layout program.

Today, users think of “customization” as being able to arrange the icons on their desktop or change a word processor’s default font. With Amber, users can take their favorite part—a styled
text part, for example—and use it in all their Amber documents. Users will like this because they will no longer have to remember how different applications handle (in this example) text. A single Amber part gives users the same capabilities and the same user interface wherever they use it.

What You Should Do

You and your company can begin preparing for Amber with these steps:

• Support the Open Scripting Architecture and make sure that your applications can both record and attach scripts (see “AppleScript, an Elemental Technology” on page 15 of this issue).

• Adopt the Bento storage model. To get more information on Bento, send your request to AMBER.IDEAS (AppleLink) or amber.ideas@applelink.apple.com (Internet).

• Consider how you might build your applications as a collection of parts.

• If you are interested in being seeded with an early version of AppleScript, send a message to the same two addresses about the kinds of innovative Amber products you’d like to create.

Apple Ventures Into Electronic Publishing

When its first personal digital assistants (PDAs) ship later this year, Apple wants to be sure users have plenty of information in electronic form for users to view with the devices. Toward this end, the company's Personal Interactive Electronics (PIE) Division has just announced that it will venture into the electronic publishing market.

As part of the new venture, Apple will create and publish its own electronic media titles as well as work with content developers to copublish or distribute their products, according to the PIE Division's announcement at June's Consumer Electronics Show in Chicago.

In addition to collaborating with developers to publish an array of electronic media titles, the PIE Division's electronic publishing strategy will be pursued in several other areas:

- the development of hardware products that will access electronic information (these are the Newton-based line of PDAs as well as a second line of multimedia PDAs to be used primarily for content viewing)
- the creation and distribution of electronic media titles
- the development of multimedia authoring tools
- expanded electronic on-line services

Said Gaston Bastiaens, vice president and general manager of the PIE Division at Apple Computer, "The electronic media market will play a key role in the success of our PDA devices. The PIE Division is looking forward to greatly expanding that market by distributing and publishing titles, providing easy-to-use tools, and collaborating with developers. We aim to support the widespread efforts of the publishing community as we work together to develop this key element of the information revolution."

The PIE Division expects to create and publish products in four primary categories: general reference, business/professional, education, and entertainment. With the development of technology, new categories of PDAs, and enhanced on-line services, publishers will have the potential to develop a completely new industry they can build from existing products.

To support developers of electronic media titles, Apple plans to release the Apple Media Kit, a professional multimedia authoring tool, later this summer. The Apple Media Kit is designed to improve title playback speed, facilitate workgroup productivity, and enable authors to deliver their titles on multiple platforms. The tool leverages off of existing technologies such as QuickTime, Apple's multimedia architecture software, and will support development for multiple platforms, including Macintosh, Windows, and future Apple consumer electronics products, including its PDAs. The Apple Media Kit is also expected to support ScriptX, the cross-platform software standard created by Kaleida Labs, Inc.

More specifics about how Apple will work with content developers on their products for use with Apple's PDAs have yet to be released. Until details are available, developers can let Apple know they're interested in the program by sending an AppleLink message to NEWTON.DEVS.

Apple Opens the UNIX Market to Macintosh Developers

Apple Computer, Inc., has unveiled plans to enable the vast majority of UNIX workstations to run, unmodified, virtually all Macintosh applications.The new technology direction, announced by Apple's Enterprise Systems Division Vice President Morris Taradalsky at the May Worldwide Developers Conference (WWDC), will for the first time give Macintosh developers access to the "open systems" market—the approximately 7 million UNIX users worldwide.

Perhaps the best part of the new technology is that it will open the new market to developers without their having to do any additional development work.

Apple will first provide the capability for IBM RISC System/6000, Hewlett-Packard Series 700, and Sun UNIX workstations to run Macintosh applications. (Apple estimates that these three kinds of systems account for nearly 90 percent of the UNIX "open systems" workstation market.) This capability will be provided by a suite of "application engines" scheduled for release this winter.

Speaking at the keynote address before 3,000 WWDC attendees, Taradalsky demonstrated working versions of the application engines, using a popular off-the-shelf Macintosh word-processing application on all three companies' UNIX workstation platforms and showing a QuickTime movie, as well.

Even though he demonstrated only one application, Taradalsky emphasized that the vast majority of developers' existing software products written for Macintosh computers using 680x0 processors will run without modification on the IBM, Hewlett-Packard, and Sun systems installed with a Macintosh application engine. (For more technical details about the application engines, see the article on page 16 of this issue.)

The application engines will be only the first in a new line of Apple products designed to enhance cross-platform capabilities and offer client-server solutions. These products, collectively called Apple Services for Open Systems, will eventually make it possible for Macintosh software services and technologies, in addition to current applications, to run on UNIX workstations and servers and X Window System™ clients.

Apple will be working closely with leading companies and standards organizations to provide Apple technologies on UNIX servers and clients. By providing these services, Apple hopes to enhance client-server solutions, moving computing resources away from centralized mainframes and toward desktop and mobile computers accessing data on mainframes and other servers.

Apple intends to provide services to open systems platforms and enable Macintosh, UNIX, and Windows/DOS clients access to those services. Apple will provide an open, published set of application program interfaces (APIs), allowing both Macintosh and UNIX developers to enhance the capabilities and ease-of-use of their products for client-server computing.

In making the announcement, Taradalsky said, “We want to bring Apple technology to more and more people. Soon, millions of users will have access to powerful, new client-server solutions with the industrial strength of UNIX and Apple’s leadership in user-centered computing.

“These services expand the market for Macintosh developers,” Taradalsky added. “They complement mainstream UNIX systems and extend the reach of Apple’s technology solutions into multiplatform client-server environments. Building on the success of our new Apple Workgroup Servers, this move will now accelerate our momentum into enterprise computing.”

Taradalsky’s statement of direction outlined three components of Apple’s new client-server cross-platform product family in addition to the application engines already described:

•*Apple Desktop Manager.* The Apple Desktop Manager for Open Systems integrates Apple desktop, file, and window management services with the desktop operating environments of open systems platforms. This technology will work with the X Window System using the Motif and Open Look window managers.

•*Apple System Services.* This component will bring a variety of previously unavailable system services to open systems platforms. These services include graphics, interface and multimedia services (QuickTime, QuickDraw GX, TrueType, localization services), networking and collaboration services (including AppleTalk, TCP/IP, SNA, and X.25), and information management services (AppleShare, Apple Open Collaboration Environment, AppleSearch, Data Access Language).

•*Apple Development Tools.* These are a set of products for developing applications that run across various platforms that support Apple Services for Open Systems. Developers will be able to deliver client-server solutions by using a combination of both Apple and UNIX services.

Joining Taradalsky on stage at the developers' conference for the announcement were executives from IBM, Hewlett-Packard, Sun, and UNIX Systems Laboratories.

Each in turn endorsed Apple's new direction.

Roel Piper, president and CEO of UNIX Systems Laboratories, who didn't attend the announcement, endorsed Apple's new direction, saying, "With this extraordinary move, Apple thrusts itself into the mainstream of multiplatform client-server computing. Apple's decision to offer its technologies to the open systems marketplace across various UNIX system platforms is precisely what customers want: the proven power, openness, and reliability of UNIX for mission-critical applications, and the flexibility to use the environment they like best. This confirms the UNIX operating system as the universal application platform."

Developers
Conference Tapes Available

The May Worldwide Developers Conference (WWDC) is over, but if you missed it or want to revisit the presentations made there about PowerPC, Amber, AppleScript, Newton, AOCE, and other Apple technologies, you can still order audiotapes and videotapes of all conference sessions.

Tapes are available from GT Recording, 101 North 85th Street, Suite 201, Seattle, WA 98103; phone (800) 878-2737; fax (206) 782-3515; AppleLink:

GTRECORDING. Contact them for an order form. Audiotapes cost $8, videotapes $30; discounts are available for multiple orders.

A list of 1993 WWDC sessions and available tapes can be found on AppleLink (path—Developer Support:Developer Services:Apple Information Sources:Developer Events:1993 Apple WWDC–Apple Developers).

QuickTime for Windows 1.1 Delivers Improved Performance

Apple Computer, Inc., has updated the Windows version of its cross-platform multimedia architecture software, QuickTime for Windows. QuickTime for Windows extends Windows 3.1 to integrate compressed video, graphics, and sound in cross-platform file formats. The latest release, QuickTime for Windows 1.1, offers improved performance and easier integration with existing Windows applications. It also provides enhanced support for developers of compression technology and audio and video cards.

QuickTime allows content developers to produce a single, hybrid CD-ROM that can play on either Macintosh or Windows computers; both platforms can play the same QuickTime movie files. QuickTime has been obtained by more than 1 million users, worldwide.

Improved Performance

QuickTime for Windows 1.1 provides significantly improved performance through support for Apple Compact Video, the advanced software compression technology introduced in QuickTime 1.5 for the Macintosh. Integration into existing Windows applications has been made easier through support for Windows 3.1 MCI (Media Control Interface) and OLE (Object Linking & Embedding) release 1.0.

As does QuickTime for Windows 1.0, the new version includes the same QuickTime movie file format, human interface, image decompression capabilities, and programming interface as the Macintosh version of QuickTime. This makes it easier for developers to implement direct support of QuickTime in Windows applications.

QuickTime for Windows 1.1 supports the advanced Apple Compact Video codec (compressor/ decompressor) introduced in QuickTime 1.5 for Macintosh. Compared with the previous QuickTime for Windows release, the codec enables movies to play either at twice the frame rate (up to 30 frames per second) or across four times the screen area (up to 320 X 240 pixel screen size, software only). The new version also supports playback through video acceleration hardware when it's used with display drivers that directly support QuickTime for Windows.

Enhanced Windows Integration

The new product's support for MCI and OLE 1.0 makes it easier to integrate QuickTime movies into existing Windows applications. The new support for MCI integrates QuickTime for Windows into authoring and presentation applications for precise playback control. With OLE support, QuickTime movies can be easily integrated into business applications that support OLE, adding impact with video footage, animation sequences, and sound clips to
documents, spreadsheets, and presentations. Microsoft's Visual Basic 2.0 is also supported for
creation of custom Windows multimedia applications using QuickTime.

Compression, Card Developer Opportunities

Support for customized add-in products enables developers to add their own compression technologies to those already included with QuickTime for Windows. For example, Intel is using this method to bundle its Indeo video decompressor with QuickTime for Windows 1.1.

In addition, QuickTime for Windows 1.1 supports a variety of audio and video cards, giving more Windows customers access to QuickTime technology. Digital video in QuickTime movies is optimized for playback on display cards capable of showing from 256 colors (8-bit) to 16 million colors (24-bit).

System Requirements and Availability

QuickTime for Windows requires a minimum of a 386SX CPU running at 20 MHz and equipped with 4MB of RAM, an 80MB hard disk, and a VGA graphics card. Movies with sound require installation of a Windows-compatible PC sound card. Also required is MS-DOS 5.0 or later and Windows 3.1 or higher. A CD-ROM drive and graphics card displaying at least 256 colors are recommended.

QuickTime for Windows is bundled with various Windows and cross-platform multimedia CD-ROM titles, clip libraries, and applications. QuickTime for Windows 1.1 is currently being seeded to developers. QuickTime for Windows 1.1 is also available to developers as a software development kit (SDK) that will be sold through APDA (see page 31 for APDA ordering information).

QuickTime for Windows 1.1 is available for licensing for redistribution with applications, titles, and media clip libraries that support QuickTime. For licensing information, contact Apple Software Licensing at (408) 974-4667; FAX (408) 862-5106; AppleLink SW.LICENSE.

New PowerBook Models Add Active Matrix Color, Greater Affordability

Apple continued to capitalize on the initial success of its PowerBook product line with the announcement of two new models, the PowerBook 180c and PowerBook 145B. These new models add improved color and aggressive entry-level pricing while maintaining the standard features of earlier PowerBook models.

PowerBook 180c

The PowerBook 180c uses a 256-color active matrix display that yields sharper images and brighter colors. The new display features:

• A wider viewing angle that allows viewers to read text and graphics without a direct line of sight

• Higher resolution (640 X 480 pixels) adding an extra 80 lines of resolution

• Support for an additional monitor that can mirror the PowerBook display or, in dual display
mode, show entirely different content

Employing a Motorola 68030 microprocessor and running at 33 MHz, the PowerBook 180c ships standard with 4 MB of RAM, expandable to 8 MB. It comes with a choice of either an 80 MB or 160 MB hard-disk drive and may also be optionally equipped with an Express Modem.

PowerBook 145B

The PowerBook 145B at $1649 (U.S. manufacturer's suggested retail for the version employing a 40 MB hard-disk drive) represents
a 25 percent reduction in the entry-level PowerBook price while
maintaining the standard features of earlier models. It features a
price/performance ratio equivalent to the PowerBook 170 at half its original price. The 145B was made more affordable through some key refinements, including:

- A standard 4MB RAM configuration, expandable to 8MB
- System 7.1 is shipped preinstalled on the hard disk
- Sound in and out remain standard, but the microphone is now optional

Both new models include a rechargeable nickel-cadium battery, AC adapter, System 7.1 software, training software, complete learning and reference documentation, and a one-year limited warranty.

The PowerBook 145B is based on the Motorola 68030 chip and runs at 25 MHz; it can be purchased with with either a 40 MB or an 80 MB hard-disk drive.

Both models are available immediately, although pricing and availability may vary worldwide. ®

Message From the Worldwide Developers Conference

By Gregg Williams, Apple Directions *Staff*

I came, I saw … I took notes—*lots* of them. I attended 27 sessions and almost filled an entire notebook. I learned a lot about the overall structures of Apple's newest technologies: PowerPC, Newton, QuickDraw GX, Amber (our compound-document architecture), and our upcoming text-to-speech and speech recognition architectures. I heard the "top-line messages" from CEO John Sculley and Apple's executive management team. (And I talked to a lot of developers, which was also a lot of fun.)

As performance artist Laurie Anderson once said, "I could write a book, and this book would be thick enough to stun an ox." If I had the time to do so, it *would* take a book for me to relate all the things I learned at the WWDC. Unfortunately, this column is all I have room and time for. (I started writing this on May 20, the Thursday after the WWDC).

Many of the technical sessions ended with a slide entitled "What You Should Do." So, since one of the purposes of this column is to give you advice you can use to plan your company's future, I thought I'd take all of these pieces of advice and try to make them into a coherent whole.

What Your Programmers Should Do

Actually, this column is different from what I think future Strategy Mosaic columns will be, largely because I'm giving concrete advice—tactics—that your engineering team should follow. Because I have so much to cover, this column is terse, but look to "Resources" sidebar on page 6 for references to other sources of information about these technologies.

Across various WWDC sessions, several common themes emerged:

• Implement Apple events and the Open Scripting Architecture (OSA) *now,* because they must be present before you can implement some of our new technologies.

• Make your code portable by following the System 7 application program interface (API) and writing code in C or C++.

• Prepare to move existing and future applications to PowerPC.

The rest of this column will go into these and other important points.

Apple Events and the OSA

First, a bit of background. The Apple event mechanism provides a way for one program to ask for something from another program (itself, a program on the same Macintosh, or even another program across a network). The Open Scripting Architecture (OSA) implements an open-ended, structured way of asking for things in the system—like referring to “the first sentence of the third paragraph of the file “Ten Ways to Survive a Company Re-org”.”

Apple Computer, Inc., has worked with developers to create standardized suites of Apple events. This way, a program can send, for example, a word-processing event like “change the selected text to bold” and know that it will be executed properly by *any* Apple event–aware program that implements the Text Events suite. AppleScript (see the Technology story on page 17) is an Apple product that gives users access to scripting Apple events; other third-party products can do similar things.

Scripting is important in its own right. We think it’s a technology that will let both developers and users create customized solutions that use Apple events to coordinate the work of multiple programs.

Amber and Voice Recognition. Scripting is not the only benefit from implementing Apple events and the OSA. It turns out that Apple’s new compound-document architecture, Amber, depends on software modules that talk to each other using Apple events and the OSA. (See the Amber news item on page 8.) You need to make your program scriptable now—not only because it’s useful today, but also because its presence will make your transition to Amber easier.

Voice recognition has been the Holy Grail (well, one of them, at least) of programmers for a very long time, and Apple is going to deliver a useful level of speaker-independent voice recognition by the end of 1993. But how do you make a spoken command control *existing* applications, which don’t know anything about voice recognition? Again, Apple events and the OSA will solve this problem; if you make your application scriptable, other software (including the voice-recognition software) can make use of it.

Achieving Scriptability. Here are the things your engineering team needs to do:

• Implement Apple events and the Open Scripting Architecture (which must be present for AppleScript to work).

• Implement the Required and Core suites of Apple events, and whatever other suites make sense for your application.

You may also want to take these additional steps:

• Make your application “recordable.” With a recordable application, users can build a script by performing the desired sequence of events and letting scripting software “record” the events as they happen.

• Add a Scripts menu to the menu bar so that users can execute useful scripts, including ones they make themselves.

Making Code Portable

Apple is moving the Macintosh onto the PowerPC platform and, later, other platforms. There are steps that you can take now to decrease the cost of moving your applications to the PowerPC and other platforms.

System 7. If you follow the System 7 interfaces and guidelines rigorously, your code will be in better shape for you to move it to PowerPC and other platforms. In particular, make sure your programs run well with System 7 virtual memory. If they do, they will probably run well under the

PowerPC implementation of virtual memory.

Use C or C++. To alter a well-known saying, “Nobody ever got fired for coding in C or C++.” Like it or not, C and, to an increasing extent, C++ are together the *lingua franca* of personal computers. Whenever a new processor or computer environment is created, the first compiler that somebody writes for it will probably be a C compiler. (Until a true C++ compiler appears for a new platform, you can use a preprocessor to convert C++ source code to C source code and then compile that.)

Here are some things you should consider doing:

• Make sure your C or C++ code follows the ANSI standard. Such code is more likely to survive the move to an entirely different environment.

• Program your future projects in C or C++. One of the advantages of C++ over C is the former’s object-oriented programming constructs. For this reason, Apple’s MacApp 3.0 and Symantec’s Bedrock application frameworks both use C++. FYI, you can use Bedrock to create 680x0 Macintosh and Microsoft Windows applications and, later, PowerPC Macintosh applications.

Use MacApp 3.0 to create 680x0 Macintosh applications and get ready for Bedrock. Apple has announced a PowerPC version of MacApp 3.0, and Symantec has announced a feature-complete developer’s version of Bedrock, both scheduled for release by the end of calendar year 1993.

• If your programmers don’t know C++, they should start learning it *now.* It is a powerful but complex language, and many people take several months or longer to learn how to use it effectively.

Regarding Programs Written in Pascal or 680x0 Assembly Language. If this is the situation for your products (and it is for many Macintosh developers), you have some tough choices ahead of you. Ideally, you should convert your program from Pascal or assembly language to C. MicroAPL offers PortAsm, which converts 680x0 assembly-language source code to PowerPC assembly-language source code. Also, Sierra Software Innovations offers p2c, which converts MPW Object Pascal (used by MacApp 1.x and 2.x) or traditional Pascal source code to C or C++ source code. See the
“Resources” sidebar for details.

Your 680x0 application will run, as is, on the forthcoming PowerPC Macintosh computers (in emulation mode), or you can convert your 680x0 object-code file—created by Pascal or any other language—to native PowerPC code using Echo Logic’s FlashPort service. (See the “Resources” section for details.) A FlashPort-converted application will run faster than an emulated one, but neither of these solutions will help you move your program to platforms other than PowerPC.

However, the installed base of 680x0 Macintosh computers is quite large. (One Apple executive quoted today’s installed base as being “over 11 million.”) You may decide that the best course for your company is to move existing products to PowerPC using one of the above methods and start using C or C++ for future products.

Macintosh on PowerPC

Much of the above material relates to the PowerPC platform, so I won't repeat it here. But make no mistake—Apple plans to move the Macintosh platform from the 680x0 to PowerPC as soon as the PowerPC chips become available; going native on the Macintosh PowerPC platform prepares you for the future. One thing your engineers need to do is learn more about the PowerPC chip architecture and how it differs from that of the 680x0 chip architecture. Then they need to examine your software and hardware products to see if any conflicts exist.

On the hardware side, don't depend on the finer points of 680x0 timing signals—the PowerPC 68020 emulator tries to match the 68020's timing but cannot do so exactly.

On the software side, be sure that your products don't depend on the presence of a floating-point unit (FPU), a memory management unit (MMU), or instruction caches—the 68020 emulator does not support them.

QuickDraw GX

QuickDraw GX doesn't replace today's QuickDraw, but it does supplement it. That way, existing programs continue to run correctly, but QuickDraw GX–savvy programs can manipulate color, graphics, and type in ways that today's programs can't. However, QuickDraw GX does represent the future of graphics on the Macintosh, and you should begin using it as soon as you can. (After all, your competitors will.)

At the WWDC, Apple engineers stated that the API for QuickDraw GX is frozen and that the first commercial version of it will ship by the end of 1993. Given both these points, you can start using QuickDraw GX now for software that you plan to ship in 1994. (The WWDC '93 CD includes the beta version of QuickDraw GX; see "Resources" for details.)

How to Get Started. To make the transition to QuickDraw GX:

• First, your applications should become QuickDraw GX-aware by supporting the QuickDraw GX printing model. This gives you the biggest "bang for the buck" and gives you cutting-edge QuickDraw GX features such as a better user interface, a versatile way for extending the printer driver's behavior, and PDD (portable digital document) files, which allows other users to view your document, even if they don't have the fonts or application you used to create it.

• Second, add support for the QuickDraw GX Clipboard, which is more versatile than the current Clipboard mechanism.

• Third, use the Line Layout Manager within QuickDraw GX for sophisticated control of type. Some features, such as context-sensitive glyph substitution, work automatically (with no extra programming on your part) when the user works with QuickDraw GX fonts.

• Fourth, implement the rest of QuickDraw GX to become "QuickDraw GX-savvy." QuickDraw GX gives you powerful tools for manipulating color, graphics, and type, and many of the ground-breaking applications of 1994 will be using QuickDraw GX.

Amber

Amber is Apple's cross-platform compound-document architecture, which will be seeded later this summer and will become commercially available in the first half of
1994. (See "Amber Provides Path to Compound Documents, Modular Software" on page 8 for more details.) Amber changes the user's task model from being application-oriented to being document-oriented. Over time, monolithic applications, which combine many separate functions, will give way to collections of part editors, each one implementing a specific function (like word processing, drawing, or spreadsheets).

The Amber design team related three steps in making the transition to Amber (assuming that your applications have already implemented Apple events and the Open Scripting Architecture):

• First, redesign your application to store documents in Bento format. (For more information
on Bento, send your request to the AppleLink address AMBER. IDEAS.)

• Second, the next time you do a major revision of your application, factor it into its functional components.

• Third, start thinking of implementing future products as Amber parts, not traditional applications.

AOCE

The Apple Open Collaboration Environment (AOCE) is important because it unifies all of a Macintosh computer's connections with the outside world. So your first task with AOCE should be to implement a mailer, which your engineers can do in several weeks of work. The mailer is a new panel added to the traditional Macintosh window; among other things, it allows users to specify the recipients of the document. Without leaving their applications, users can then send the document via AOCE. By implementing mailers, you make your product open to the rest of the AOCE universe. It’s a good investment of your time. (For more on AOCE, see the page1 story on AOCE in the March 1993 issue of *Apple Direct.*)

Newton

The first Newton product (as yet unnamed) will ship sometime this summer, and development tools will be available by the time the first product ships. Apple’s Personal Interactive Electronics (PIE) division will open software development and support to everyone. (See “Newton Is Open for Development” in the June 1993 issue of *Apple Direct.*)

If you haven’t done so already, send your name and address to the Newton team at the AppleLink address NEWTON.DEVS; you may want to add your thoughts on the type of Newton software you’d like to develop. The Newton group will then send you developer support information as it becomes available.

Apple’s Developer University will be offering Newton software development courses beginning in August. Contact them at (408) 974-6012 (in the United States) or send a request to them at the AppleLink address DEVUNIV.

PowerOpen

PowerOpen is an alternative operating system based on the venerable UNIX operating system. (System 7 will remain the operating system that will ship with every PowerPC Macintosh.) PowerPC Macintosh computers running PowerOpen will run most existing Macintosh programs. (See “PowerOpen Association Announced” in the May 1993 issue of *Apple Direct*). Compatibility problems can occur, though, and one way to help make your applications ready for PowerOpen is to make sure they work properly on a Macintosh running the A/UX 3.0 operating system.

If you take the additional step of converting your application to native PowerPC code, it will also be ready for PowerOpen and will run faster than your original 680x0 version.

Text to Speech

Beta versions of Apple’s new text-to-speech technologies are included on the WWDC ’93 CD (see the “Resources” sidebar). One of them, tentatively named *MacinTalk 2.0,* works on a Macintosh Plus or later model. The second, which gives better speech quality, is code named *Gala Tea;* Apple recommends that you use it with a Macintosh LC or later model. Now’s the time to try these technologies out and start thinking about how they might enhance your products.

Internationalization

Two things you can do now will help make your applications easier to localize and more suitable for the global community. First, make sure your applications support System 7.1 and WorldScript. Second, start thinking about using the Unicode text standard for future products. (By the way, the first Newton device uses Unicode for all occurrences of text.)

Making Choices

Your company almost certainly doesn’t have the resources to implement all the changes listed in this column. But with the above information, you can make more informed decisions about your future direction and the implications of these changes. Certainly, you should implement Apple events and the Open Scripting Architecture, because many upcoming technologies depend on them.

You have some choices to make, but that’s not entirely a bad thing. Apple’s upcoming technologies are designed to give you an opportunity to enhance your current products and create entirely new ones. I hope that this column will help you make those choices and find those opportunities.

As part of the Apple Developer Group’s charter to help developers be successful in their businesses, Strategy Mosaic is a monthly look at important pieces of Apple’s overall strategy. Our intent is to give you information that you can use to plan your company’s future. This column is based on information from managers throughout Apple Computer, Inc. Gregg Williams has been writing about Apple technology since 1982. You can contact him at GREGGW on AppleLink, or by phone at (408) 974-3264 (in the United States).

****

Resources

The WWDC ’93 CD

Apple shipped this CD in mid-May to Apple Associates and Partners. It includes both code and documentation on the following subjects: Apple Open Collaboration Environment (beta), AppleScript, Apple Shared Library Manager, ColorSync, Japanese Language Kit, LaserWriter 8.0, Apple events, PowerPC, QuickDraw GX (beta), QuickTime 1.6, QuickTime for Windows 1.1 (beta), SCSI Manager 3.0 (beta), Sound Manager 3.0 (beta), and Text-to-Speech (beta).

The Developer CD Series

This series of CD-ROM discs, sent monthly as part of the Technical Information Mailing, contain (literally) megabytes of documentation, software, and sample code. Check it for the latest information on a subject.

AppleScript, Apple Events, and the Open Scripting Architecture

• APDA offers an Apple Events/AppleScript Programming Tutorial ($150, part number R0224LL/A).

• See the Apple Events folder in the May 1993 Developer CD (pathname—Dev.CD May 93:Tool Chest:OS/Toolbox:Apple Events). To learn more about how to add Apple events and event suites, see the documents in the “Beyond the Required Suite” folder and read the *Apple Event Registry* document.

• See the “AppleScript Runtime” and the “New Apple event suites” folders on the WWDC ’93 CD.

• See “Better Apple Event Coding Through Objects,” by Eric M. Berdahl, in *develop* magazine, issue 12 (December 1992).

PowerPC

• The FlashPort technology for converting 680x0 code to PowerPC code is available from Echo Logic. Their phone numbers in the United States are (908) 946-1100 (voice) and (908) 946-9146 (fax).

• The p2c source code translator is available from Sierra Software Innovations. Their phone numbers in the United States are (702) 832-0300 (phone) and (702) 832-7753 (fax).

• The portAsm source code translator is available from MicroAPL Ltd. Their phone numbers in Great Britain are (+44) 71 922 8866 (phone) and (+44) 71 928 1006. They are also available on AppleLink at address MICROAPL.

• See “Apple Event Objects and You,” by Richard Clark, in *develop* magazine, Issue 10 (May 1992).

• See “Macintosh on PowerPC: Top Developer Q&As” and “Preparing for PowerPC: Ten Commandments” in *Apple Directions,* June 1993.

QuickDraw GX

• See “Print Hints: Looking Ahead to QuickDraw GX,” by Pete Alexander, in *develop* magazine, Issue 13 (March 1993).

Internationalization

• See “Writing Localizable Applications,” by Joseph Ternasky and Bryan K. Ressler, in *develop* magazine, Issue 14 (June 1993).

• See *Guide to Macintosh Software Localization*, Addison Wesley (publisher), 1992.

Keeping You Up To Date

Last month, we promised that *Apple Directions* would provide new kinds of material and new voices.

This month, two of those new voices make their debuts: Amanda Hixson, who recently ended her long tenure at Apple to become a journalist and consultant, delivers the first installment of a new feature, IndustryWatch: News & Perspective. And Don Norman, who recently joined Steve Wozniak and Alan Kay as an Apple Fellow, makes his first appearance on Page 12.

First, you might be asking Why IndustryWatch and why Amanda? They're part of *Apple Directions'* purpose of supporting development firms' overall business proposition. If we're to live up to that promise, we have to let you know about what's going in the personal computer industry, both in and out of Apple. IndustryWatch will tell you about goings on in the personal computer industry beyond Apple's walls.

Needless to say, there are many publications that already do that, but none of those others provide what only *Apple Directions* can provide: first and foremost, we give you the "official" Apple word about our own products and business strategy; second, and just as important, we offer unique perspective for *Apple* developers. IndustryWatch fits into this second category.

IndustryWatch can't give you all the news about the personal computing industry, but each month Amanda will be talking to insiders to find out what's behind the news and digesting the month's most interesting material. She'll report just what she thinks Apple developers need to know and tell you what's important about it.

Amanda's been around, to say the least, both in and out of Apple. During her 5-plus years here at Apple, she's been an evangelist, a marketing manager, and an analyst; her last assignment, for the past three years, was engineering project manager for QuickDraw GX. Before coming to Apple, she covered the personal computer waterfront for a number of influential publications.

What distinguishes Amanda is her reputation for calling them as she sees them and telling the truth; she's also a strong "keeper of the Apple faith" (actually a title she held for brief time at Apple). We think she's uniquely qualified to keep you abreast of goings on in the industry.

Professor Donald Norman should need little introduction. His writing about human, user-centered use of technology in books like *The Design of Everyday Things* is well known and has been very influential in Macintosh hardware and software design. His purpose in *Apple Directions:* to help us all remember to put the user first and to push us toward ever-greater and more humanly useful applications of our wonderful technology. He'll do this by periodically answering readers' questions.

Apple Directions **on AppleLink**

Apple Directions (and before that *Apple Direct*) has always been as timely as possible, but the limitations of paper-based distribution sometimes mean readers have to wait more than they (or we) would like to get their monthly issue.

To change that, we're making electronic files for articles in *Apple Directions* available on AppleLink about *two weeks before the printed version even goes into the mail.* This means readers (especially those outside the U.S.) won't have to wait so long to read about the latest goings on.

Starting July 1, we'll post preliminary, unformatted *Apple Directions* content on AppleLink (path—Developer Support:Developer Services:Periodicals:Apple Directions). On the first of each month, you can look in the *Apple Directions* folder at that location for new material. Then, on the fifteenth of each month, we'll replace that material with finalized content and whatever stories were too late-breaking to make it into the first posting. (If those dates fall on a weekend, the material will be posted the Monday after.)

And speaking of AppleLink, last month we printed an incorrect address for *Apple Directions.* The right address is A.DIRECTIONS; use it early and often to let us know what you think our new publication.

Paul Dreyfus
Editor

True Stories: NT, ATMs, and Technotainment

By Amanda Hixson

[IndustryWatch: News & Perspective is a new feature of Apple Directions. *It provides perspective for Apple developers about news and events in the personal computer industry beyond Apple's walls. IndustryWatch and its author are introduced in this month's Editor's Note; see page 2.]*

Microsoft NT Unveiled…Kind Of

Well, the big day has come and gone. May 24, 1993, was the day we all expected Microsoft to unveil Windows NT, computerdom's rumored Rosetta Stone, the "mother" of all operating systems. Instead of the big Kahuna, Microsoft handed out press releases and rain checks and told oglers to come back and see the real product in a couple of months. That's right, another announcement to tell us when to expect the next announcement.

Why is it that it's so easy to get sucked into thinking new technologies are going to solve all our problems, basing our expectations on rumors and promises? We chase rainbows and continue to search for easy answers and instant wins. Believe me, I know. I've spent a couple thousand dollars on stamps for Publishers Clearing House–type sweepstakes and tickets for million-wins in the lottery. Have you ever actually met a sweepstakes or lottery winner? I haven't.

There is no panacea for computing's multiplicity of problems. No company is going to suddenly show up with a shrink-wrapped cure-all for computing's woes. If we all believed the hype handed out by software and hardware vendors about impending products, we'd be paralyzed by our own indecision (and, admittedly, we sometimes are). Face it, none of us is really going to put all our eggs in one basket.

So why is it that an entire industry can appear to hold its breath waiting for "the" magic product? After years in development, Windows NT has reached the mythical proportions of such a product. Does any of us expect it, or any other recently announced products, to really live up to such expectations?

Sure, Microsoft has *shipped* a zillion copies of Windows, DOS, and their brethren. Yes, it's true, they have created an impressive ground swell among some big names such as NCR, DEC, Mips, NEC, and others for the impending

Windows NT release (now slated for the end of July). Most of those companies, however, aren't looking at NT as their savior in all computing markets. They are targeting the same market Microsoft is with NT: that is, high-end, networked, client-server, mission-critical solution seekers. Microsoft anticipates NT sales of a million units during its first twelve months on the market.

STOP, CLICK, REWIND

A million units?

If Apple were to make that low an annual projection for its next greatest operating system release, developers might greet it with laughter, for good reason.

Just how many $18,000, Pentium-based (Intel's new, literally red-hot microprocessor), symmetric multiprocessing servers can any company sell to this lucky million? Unless one is assuming that all the new Windows NT customers will rush out to pick up a six-pack of hardware when they buy their NT-ready Windows applications, my guess is not that many. Even if a handful of vendors strike it rich, what happens to the dozens that won't? They keep licking stamps.

Sure, for companies that consider themselves fortunate to sell five or ten thousand copies of anything to their high-end customers, a million potential customers sounds pretty good. But for those of you selling thousands a month, how attractive is that market?

To make sure you can really take advantage of NT, it also appears that you really need to support other Microsoft architectural technologies such as Open Database Connectivity (ODBC), its Mail API (MAPI), the Windows Open Services Architecture (WOSA), object linking and embedding (OLE), DOC, SLEEPY, and GRUMPY, just to name a few.

How much faith does Microsoft have in a solution it wants you to bank on when rumor has it that you'll also need to deal with a new 32-bit version of Windows, code-named *Cairo* (only partially based on the Windows NT kernel, according to one trade journal), that's expected to outsell NT on the desktop? Its key feature, according to spokespersons from Microsoft: a Component Object Model (can you say Open Scripting Architecture and Amber?). *[Note: See pages 5-–9 and 17-21 of this issue for more about these* Apple *technologies.]*

And don't forget there are other interim version of Windows, one code named *Chicago* that (I think I have this straight) bridges Windows 3.1 and Cairo, but lacks some of Cairo's glitzy object-oriented features. To also help solidify the foundation Microsoft hopes to lay, the company is purportedly taking a long, hard look at other current building blocks, such as the not-so-robust Modular Windows technology used in several multimedia ventures.

Toss in the the fact that NT is just now being tested by real corporate users, and you have to ask yourself if anyone will really want to put all their eggs in that particular basket, especially when it doesn't appear to the casual observer that Microsoft is.

Don't think I'm excusing Apple or any other software vendor from similar hype, because I'm not. Everybody does it. What I am saying is that you need to consider the trade-offs and benefits of any potential development environment before committing huge amounts of resources to it—unless, of course, you have huge amounts of resources to waste.

I will guarantee you one thing: Many of the folks jumping on the Windows NT bandwagon haven't dropped development for other environments. I've talked to quite a few folks who are patiently waiting to judge the success of PowerOpen (the new version of the UNIX® operating system based on Apple's A/UX 3.0 and IBM's AIX, under codevelopment by both companies), keeping resources in reserve for whatever Taligent is brewing, and waiting to see what Macintosh- (and other operating system-based) PowerPC RISC computers can do. And believe me, nobody is writing off OS/2, UNIX (in all its flavors), and various other long-term development platforms in which they have major investments, just yet.

And We Think People Go Out of Their Way to Pirate Software

For those of you who saw *Terminator 2, Judgment Day,* I'm sure you recall the scene in which the young John Connor slips his secret decoder card into an automated teller machine (ATM), waits as it comes up with a password, and scams a couple of hundred dollars for himself and his pal. “Easy money,” as John said in the film.

That scenario should have sent shivers up and down your spine. Could it really be possible that it might one day be easier to steal funds from the bank accounts of innocent citizens than it is to copy software? If people with money aren't paying for your products now, how can you expect them to purchase anything if their accounts are vacuumed clean?

Whoa! Earth to Amanda, wasn't that scene a little far-fetched? I think not, but will let you judge for yourself.

A few weeks back, some men allegedly pretending to be bank representatives asked if they could install an ATM in a Connecticut shopping mall. Before the folks representing the mall could get back to the men, they showed up with an ATM and installed it near some ATMs belonging to a real financial institution.

One of the two honest machines was allegedly disabled by the perpetrators and the bogus machine turned on. Folks waiting for the real ATMs discovered that one was out of service and, as many of us would have, tried the bogus machine instead.

After a card was inserted, the machine apparently acted normally until the card's owner punched in a personal identification number (PIN). Instead of processing the requested transaction, the bogus machine apparently recorded both the card number and PIN, flashed a "temporarily out of order" message on the display, and then spit the card back out.

The phony bank representatives later returned, removed the bogus machine, and then used the recorded card numbers and PINs to slip an estimated $50,000 out of the accounts belonging to the people who had tried to use the phony ATM.

Multimedia Maastricht?

Is it just me or have some of you noticed that, lately, alliances and technology partnerships have been appearing faster than bugs at build time? Telecommunications giants are forming partnerships with monolithic computer companies; Hollywood heavies are hanging out with multimedia homeboys from Silicon Gulch and Silicon Valley in settings right out of the "Wild Palms" miniseries.

Apparently many companies are in agreement that fiber optics, telephony, communications, satellite broadcasts, high definition television (HDTV), consumer electronics, entertainment (film, video, music), on-line services, and a slew of other stuff are all ultimately going to slam together to form a new industry (or maybe a black hole). As Rich Frank, studio president of Walt Disney Co., was recently quoted as saying in *The Wall Street Journal* about this new industry, "If you're not part of the steamroller, you're part of the road."

The way things are going, all these industries will need a constitution for union similar to the treaty signed 18 months ago in Maastricht, Denmark, as a Constitution for European Union. Considering the divergent needs, skills, and markets of some of these companies, it could take about as long for the high-tech and entertainment industries to get this new technocracy together as Europe has spent in vain attempts at unity since World War II.

Just as in Europe, a key issue that the new technocracy must address is the determination of a common currency. It seems pretty evident to me that content is going to be a heavy component of that coming currency standard. Unfortunately, it obviously isn't all that evident to some of the folks entering into these partnerships—a fact that can have an impact on everyone involved in the impending technotainment industry, including you.

Take the recent IBM/Blockbuster Entertainment announcement in which the two companies disclosed a joint venture named *Fairway Technology Associates.* The deal involves IBM's Fireworks Partners unit and Blockbuster's Newleaf Entertainment Corp. subsidiary.

The plan is to have Fairway deliver just-in-time multimedia, such as CDs (and maybe software, games, and other related technologies), at the point of sale: In the case of CDs, a CD press in a kiosk at the local music store would generate CDs on demand. Sounds pretty nifty to me, and it makes sense if you consider the number of titles not usually in stock that you might want to purchase.

According to various accounts of the IBM/Blockbuster story, major owners of music copyrights—companies such as MCA Music, Warner Music, and Sony Music—weren't so thrilled by the IBM/Blockbuster high-glitz announcement. Apparently IBM and Blockbuster Entertainment forgot to take into account the fact that they don't own copyrights to (and licenses to distribute) the information their technology will be designed to duplicate, at least not in the case of much of the music currently residing on CDs. Worse yet, they apparently forgot to mention to the record companies that do own the content that they were even making the announcement.

OOPS!

As a former high-ranking Apple officer used to say, "God is in the details." That goes beyond product features and programming magic. Every aspect of product development and delivery must be scrutinized. Remember to cross every "t" and to dot every "i," beginning with the original product concept documents to final delivery and aftermarket support agreements.

• • •

As I hope you can see from this month's content, this column is written in a fun way to help broaden your understanding of technical, societal, and strategic events that affect developers. I look for information and stories that offer lessons from which we can all benefit, and I hope they help you wade through the bewildering array of information our industry generates every day.

I encourage you to respond to these stories. I'd also like to hear from you about other fun, weird, serious, strategic, and just plain interesting events that other developers need to know about. You can reach me on AppleLink at A.HIXSON.
®

What Are People Really Going To Do With Computers?

Some months ago, I found myself in the office of Don Norman, the most recently anointed Apple Fellow, asking him if he'd like to help inspire developers to even higher heights of development by writing for Apple Directions. *He had a great idea (isn't that what we have Apple Fellows for?): being too busy to write regularly, he suggested that I ask developers to send him questions by AppleLink. He'd mull them over, and when he got time he'd pick one or two intriguing ones and answer them. I posed the above question to him, and here's his answer.*

—Editor

A long time ago you asked me, "What are people really going to do with computers?" I now find myself on the Amtrak train traveling between New York and Washington with another two hours to go, so I finally have time to respond.

Keep sending questions. My rather busy life at Apple doesn't leave me much time to respond, so I can't guarantee I will always get back to you, but I can guarantee I will try. I'll carry the questions around with me in my PowerBook Duo (oops, I mean in my portable writing tablet), and when inspiration and free time manage to converge, why, I will provide an answer.

The answer: Nothing. Or everything. It all depends upon what you mean by *computers.*

Let me rephrase the question: What should the everyday person do with computers?

By *everyday,* I mean the vast number of noncomputer, nontechnical folk out there in the world, including us professional programmers, scientists, and engineers when we aren't sitting hunched up over our keyboards but, instead, during those rare instances when we are trying to live normal lives.

The answer is NOTHING. NOTHING at all. In fact, the question is badly phrased: ILLEGAL QUESTION TYPE 306#.

Look, the computer should be a tool, not the end point. Has anyone ever asked you what you do with everyday tools? What are people really going to do with their pencils? Or with their shovels, pliers, cars, telephones . . . ? Put the question that way and you see how silly it becomes. What do we do with our pencils? We write, draw, twiddle, suck, and scratch with them: We use them when we need them to help accomplish the tasks we want to do.

See what I mean: Bad question. I look forward to the day when we won't want to use computers, but instead we will want to write letters, change the contents of our newspapers, ask for the movie "Terminator XII" to be shown at 9:30 P.M. instead of the 8:00 originally requested.

Or suppose I want to refurbish my living room. I might turn on my wall screen and ask to see what couches are available at the local department store, then ask to see them, one at a time, displayed in a picture of my living room. ("Hmm, if we moved the couch over against that wall, and moved the bookcase over there . . . .")

Now, all these activities require computers—in fact, a lot of computational power—but I would never even have to realize that I was using my computer. I would just do my tasks.

So what will people use all those computers for? If by *computers* you mean those monsters that take so much space on our desks today, the answer should be nothing at all. In fact, those computers will hardly exist.

But if by *computers* you mean the hidden processors that make those new activities of the future possible, then the answer is probably more like, "Why, for almost everything."

Professor Norman joined Apple in January 1993 as an Apple Fellow. Before life at Apple he was professor and chair of the Department of Cognitive Science at the University of California, San Diego. He is the author of *The Design of Everyday Things.* His most recent book is *Things That Make Us Smart* (Addison-Wesley, 1993). Send your questions to him at A.DIRECTIONS; he'll answer them as his time allows.

Tech Notes Changes: Numbers Are Back

Due to popular demand, we've again given Macintosh Technical Notes numbers, in addition to their titles.

Numbers have been added to the titles for ease of reference and updating. Apple's Developer Technical Support engineers figure it will be easier to refer to "Hardware Tech Note #25" instead of "M.HW.MEMCONFIGS."

The numbers should also make it easier for you to remember which Tech Notes you need to use; once you've learned the numbers of commonly used Tech Notes, you'll be able to remember them by that number instead of (or in addition to) their longer title.

The new Tech Note numbers have no relation to the previous numbers, which were eliminated a year ago in favor of descriptive titles. Instead, Tech Notes are numbered within each of the 16 Tech Note categories. That is, notes in the Communications category receive numbers from 0 to 1500, as do those in the Devices, Files, and other categories listed in the table at the end of this article.

Tech Note 0 within each category describes what's new in that category and the contents of the category. Standard Tech Notes receive a number from 1 to 500, Q&As are numbered from 501 to 1000 within each category, and *Inside Macintosh* errata notes are numbered 1,001 to 1500.

You can still find the Tech Notes you're looking for (and view them using Apple DocViewer) by title, just as you have for the past year. Designations follow the convention *T.DC.ItemName,* where *T* is the general technical category (such as Macintosh, Apple II, or RISC), *DC* is the documentation category (the categories listed below), and *ItemName* is a very short description of the note's contents.

For example, the "Using a Purge Proc" Tech Note carries the designation M.ME.PurgeProc (where *M* means *Macintosh* and *ME* means *Memory*) as well as ME 11 (Tech Note 11 in the Memory category). "VCBs and Drive Numbers: The Real Story" is designated M.FL.VCBandDrive Num (*M* means *Macintosh, FL* means *Files)* and FL 34 (Tech Note 34 in Files).

Let us know what you think of the new numbering scheme, and any other aspects of Tech Notes, by sending an AppleLink message to TECH.NOTE.

****

Macintosh Technical Note Categories

Communications (CM)
Devices (DV)
Files (FL)
Hardware (HW)
Interapplication Communication (IC)
Memory (ME)
Operating System (OS)
Overview (OV)
Platforms & Tools (PT)
Printing (PR)
Processes (PS)
QuickDraw (QD)
QuickTime (QT)
Networking (NW)
Text (TE)
Toolbox (TB)

System Software Edition, July 1993

July is here, which means it’s time for another System Software Edition of the Developer CD Series. This month’s disc contains over 140 MB of new and revised system software, enablers, and extensions, including golden master versions of Korean (Hangul), Chinese Simplified, and Chinese Traditional system software version 7.1; several new localized versions of QuickTime 1.5; and a new System Enabler (1.3.1) that works with the forthcoming PowerBook 180c as well as the PowerBook 160, PowerBook 165c, and PowerBook 180.

This month in the What’s New? folder we have almost 50 MB of new and revised technical documentation and tools, including the ones listed below.

Developer Notes

Included here are developer notes for several soon-to-be-released products, including the Macintosh PowerBook 145B and 180c and the LaserWriter Select 310.

Human Interface Notes

Human Interface Notes are a collection of human interface guidelines for Macintosh applications. The introduction to these notes has been updated to provide more accurate information about the Human Interface Notes and Guidelines and contact information for the human interface groups at Apple.

Inside Macintosh: QuickTime Components

Describes how to use and develop QuickTime components, such as image compressors, movie controllers, sequence grabbers, and video digitizers.

International Tech Notes

The first of a new series, “Internationalization Checklist” is a document discussing internationalization issues that may arise in the development of Macintosh software.

Macintosh Easy Open Overview

The What’s New? folder includes several documents relating to Macintosh Easy Open, an extension to System 7 incorporating the Translation Manager application program interface (API), which extends the Macintosh architecture to provide implicit file translation when a document is opened. Designed to simplify access to information, Macintosh Easy Open makes translating documents as easy as any other basic Macintosh task.

AppleGlot 2.0

AppleGlot version 2.0 is a text translation tool that succeeds AppleGlot 1.1. The 2.0 version is batch oriented, while AppleGlot 1.1 was single-file oriented. This version features batch operation, background processing, and Resorcerer template model support.

LaserWriter 8.0 Driver

This is the driver for the current printing architecture that many have been waiting for, with lots of nifty new features. Features include a streamlined user interface; support for PostScript™ printer definition (PPD) files for extensive printer-specific features; the ability to save EPS files; new API routines that assist in QuickDraw-to-PostScript translation; the generation of PostScript Level 2 code for Level 2 devices; extreme compatibility; solutions to many problems with older drivers; and faster performance, especially in the background.

QuickTime 1.6

The QuickTime system software extension adds capabilities that let your applications integrate graphics, audio, video, and animation into documents. By providing a standard way for all Macintosh programs to control these multimedia elements, QuickTime makes them easier to use. QuickTime 1.6 includes the following new features:

- smaller size
- conversion of audio tracks to movies
- Macintosh Easy Open support
- ColorSync support
- improved grayscale
- smoother text

Virtual User Tools

The What’s New? folder also contains two new external tools to be used with Virtual User 2.0. The first tool, Ivy, provides image-checking capabilities: It compares two images and returns information on whether the images are the same or different. The second tool, Memory Monitor, checks memory usage and processes running on a target machine: It reports heap check information, a list of all the running processes, About This Macintosh information, and so on.

In the What's New? folder, you'll also find updates to Macintosh Tech Notes and the Apple Bug Reporter and C/F Registration Requests stacks.

In next month's Tool Chest issue of the Developer CD Series, we hope to include a new version of ColorSync, the Drag Manager, more Language Kits, a way cool network game demo, and more tools. See you there!

Alex Dosher
Acting Developer CD Project Manager

Secrets of the Macintosh Interface

By Peter Bickford

“So why hasn’t Apple fixed that ‘dragging a disk to the Trash to erase it’ problem yet? I mean, they’ve had years *to get around to it!”*
— Anonymous heckler at the InterCHI conference in Amsterdam

“Something I’ve been wondering about for a long time: Why did you [Apple] go with Chicago as your standard font in the first place? I mean…it’s not the best-looking font.…”
— A more courteous heckler, in a hall conversation, on the first day of the Worldwide Developer’s Conference

First off, an apology to my regular readers: I promised that I’d be answering letters this month, and the above comments don’t really qualify as such. I hope that my using this month’s column to answer these rabble-rousers doesn’t discourage you from sending in your usual, well-reasoned interface questions. It’s just that I’m beginning to suspect that I’ll never get a moment’s peace at an industry conference unless I put questions like these to rest.

Yes, it’s time to throw off the blanket of secrecy and tell all. Although I swore the sacred blood oath of silence upon first joining Apple, I just can’t take it anymore. So unless the interface police get me first, this month’s column will reveal the answer to not only these, but also to many of the other imponderable questions of the Macintosh interface. And while much of what follows may seem weird, humorous, or downright silly, it may provide you with some valuable lessons on designing interfaces.

What’s the Deal With the Chicago Font, Anyway?

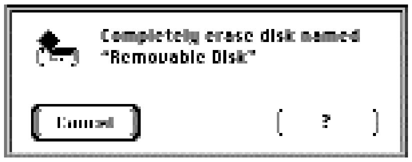
You’d think that Apple, of all companies, would have a clue when it comes to graphic design. Apple is, after all, the company that made “desktop publishing” a household word. In everything from our manuals to our advertisements, we try hard to give the impression of elegance and sophistication. But then, there’s that pesky Chicago font.

By typographic standards, Chicago is an impressively horrible font (as you can see from its intrusion into this otherwise attractive article). It’s so heavy that

using it for any large amount of text makes the entire page hard to read. More to the point, it's exactly the sort of
funky, weird-looking font that you should avoid if you want people to take what you write seriously.

Why then, did we decide to use Chicago at all, much less make it the standard system font? The secret is this: The original Lisa/Macintosh interface showed a disabled button by drawing its name with every other pixel missing. That gave a nice "dimmed" effect, even on the original, black-and-white Macintosh. However, if you wanted a user to be able to read the name of the dimmed button, you'd have to start with a very thick font. Thus was born Chicago—a font hand-tuned for just such abuse.

Recently, Apple started using a more elegant method of dimming controls, drawing them in actual gray, whenever possible. This results in a much nicer effect, and would ultimately seem to pave the way for us to abandon our friend Chicago. However, millions of black and white Macintosh computers can't take advantage of this new trick, so Chicago is still the font of choice for keeping disabled controls legible.



Completely erase disk named
"Removable Disk"

I'm OK, You're OK

OK and Cancel buttons have been a standard feature of dialog boxes practically since the dawn of Macintosh time (which, as far as the original Macintosh system clock was concerned, occurred at midnight on January 1, 1984). You might be surprised, then, to learn that the original designers of the Lisa/Macintosh interface didn't want to use OK as a button name at all.

Instead, in dialogs where clicking one button would mean "Go ahead and do it" and the other button would cancel, the original designers wanted to use *Do It* as the name of the button that would, well, *do it.* The only problem with this very sensible approach was that in the Chicago font, an uppercase *I* looked exactly like a lowercase *l*. Moreover, since the Lisa used proportional fonts, the space character between the *Do* and the *It* seemed rather small. As a result, users read *Do It* as *DoIt* and were understandably confused, even offended. Reluctantly, the designers adopted *OK* as an alternative.

Cache as Cache Can

The saga of the disk cache should serve as a cautionary tale. It warns of the importance of never trying to pull a fast one on your users, and of the lengths that users will go in their attempts to make sense of an interface.

As any good computer scientist knows, a disk cache is an area of memory devoted to holding information that has been read from the disk. Whenever a computer needs information, it usually saves time to first see if it can find the information in the cache, rather than having to do the much slower work of pulling it off the disk.

By using a control panel, Macintosh users can set the size of their machine's disk cache. In addition, users of systems prior to System 7 had a set of radio buttons to turn the cache on or off. Unfortunately, I, like most users, was pretty hard-pressed to see much of a difference one way or another. So, being a trusting soul, I devoted whatever the standard amount was to my cache and imagined that my machine must be much slower without it. Later, at least one respected magazine issued an extensive report on the subject of cache setting. Its major conclusion was that even the minimal cache was better than no cache at all, and that the big caches were generally better than small caches.

Many years later, during the development of System 7, I noticed that the Memory control panel no longer had the radio buttons for turning the cache on and off. When I asked why they'd been left out, I learned a shocking truth: They'd never really worked in the first place. In effect, the cache was always on, and all "turning the cache off" did was move it down to the minimum size.

All of this, of course, was fixed in System 7, and the Memory control panel now sports the small message *Always On* in place of the old controls for turning the disk cache on and off. But lying to your users with your interface always has a price; our technical support line now logs calls from concerned users wondering why they can't turn their disk cache off anymore since upgrading to System 7!

That Disk/Trash Thing ...

It took me about three years to convince my mother that her Apple II was no longer the computer of choice for running her home business. Ultimately, I had to buy her a Macintosh, trick her into sending me her Apple II diskettes, and finally journey to Colorado for a session of hand holding and computer tutoring. I must have said, “Don’t worry, the Macintosh is really easy” about a hundred times before her blood pressure started to approach a normal range. When we were finished, the magic of computers had transferred her ancient Apple II files onto a modern, 800K floppy disk.

Then, smiling with confidence, I dragged the disk to the Trash. Mom (naturally) freaked.

Exercise for the reader: Try explaining to your mother that dragging files to the Trash erases them forever, but dragging a disk to the Trash safely ejects it. Explain further that you’re one of those interface designer folks who came up with this sort of thing.

My explanation went like this: When the original Macintosh came out, it had one floppy drive, and no hard disk. So, when a user wanted to copy information from one disk to another, you needed to be able to show one icon to represent the disk that was in the drive, and a “ghost image” of the disk to be copied to. The engineers decided to handle this problem by using the Eject Disk menu command to eject the disk, leaving its image on the desktop and the “drag to the Trash” gesture to both eject the disk and *throw away its image.*

What may have seemed like a necessary evil in the days of no hard disks is now perhaps the most legendary embarrassment of the Macintosh interface. And, if only to avoid catching grief at conferences, I admit it’s one feature of the Macintosh interface that we’d all like
to change.

During System 7’s development, we added the Put Away command to the Special menu, and thought we could use this to be rid of the Trash-eject behavior forever. But when we removed the trash-eject behavior, we were bombarded with complaints from users who had come to learn, and rely on, this interface quirk. Embarrassing as it was, we began to realize that we could not get rid of it, unless we were willing to disrupt literally millions of users. So sadly, the Trash-eject stayed.

The moral of the story: If you’ve got a bad interface design, it’s better to fix it in the first release. By the time you get around to revising it, it may be too late.

Until next time,
—Doc
(AppleLink: THE.DOKTOR)

Peter Bickford is a member of the Human Interface Group in Apple’s Enterprise Systems Division. Pete’s on sabbatical leave right now, hiding out from the interface police and collecting more real-life metaphors.

Macintosh Application Engines for UNIX: Taking Your Applications Into the Open Systems Market

At May's Worldwide Developers Conference, Apple announced a new technology direction coming out of its Enterprise Systems Division business unit: Apple Services for Open Systems, which will allow existing Macintosh applications to "go cross-platform" and run—without modification—on UNIX workstations and servers and X Window System clients.

The first products in this area will be a suite of "application engines" for UNIX to be released beginning this winter. The Macintosh application engines will let Sun, Hewlett-Packard, and IBM RS-6000 UNIX workstations run, as is, applications written for Macintosh 680x0 computers (see related story on page 9).

We expect that most developers are curious to know if their applications will work with the new application engines. To begin to address these issues, Apple's Open Systems Group has put together a brief set of "compatibility guidelines." We've reprinted these guidelines below; Apple Directions *will provide more details about Apple Services for Open Systems when they're available.*

—Editor

Macintosh Application Engines for UNIX Compatibility Guidelines

The Macintosh application engines for UNIX put a Macintosh desktop in an X Window System window giving UNIX workstation users access to Macintosh applications.

In future releases, this technology will provide improved access to underlying Macintosh services.

Developer Benefits. The new technology brings developers two important benefits:

- access to a new customer base for current applications
- minimal work required for compatibility—most cases require testing only

How Do They Work? Apple has rewritten portions of the Macintosh Toolbox to run on the native RISC platform. Apple has also developed a mixed-mode switcher, which moves between the native and emulated portions of the Toolbox. The engine handles the switching, providing near-native RISC performance for parts of your application.

Can You Use the Application Engines? Applications that conform to the guidelines in *Inside Macintosh*, Volume VI (pages 3-24 to 3-28), and run successfully under the A/UX Toolbox (version 3.0 or later) should be able to run under the application engines. (Note: *Inside Macintosh* documents releases of A/UX previous to A/UX 3.0.) The application engines are compatible with System 7.1.

Some Guidelines. Developers should be aware of these compatibility guidelines if they'd like their applications to run under the new technology:

• Applications should include a 'SIZE' resource and call the WaitNextEvent function in the main event loop.

• When running under the engines, some low-memory global variables don't work the same
way as they do in the Macintosh Operating System.

• Applications must be 32-bit clean.

• Before relying on features that may not be present, applications should use the Gestalt Manager to determine which versions of managers and drivers are present in the current operating environment.

• 680x0-FPU instructions are not supported by the new technology. Applications should use SANE instead.

• QuickTime features will not be supported in the engines' first release.

• Applications should use QuickDraw instead of drawing directly to the frame buffer.

• Applications cannot manipulate hardware directly.

• Applications should not use privileged 680x0 instructions. The restrictions are the same as for A/UX.

• The Toolbox managers listed in "Managers Not Supported" are not supported or are partially supported by the first releases of the Macintosh application engines (or under A/UX).

For More Information. For more information on the Macintosh application engines for UNIX, contact Bill Convis, Apple Open Systems Evangelist; AppleLink: BILL.CONVIS.
****

Managers Not Supported

Alias Manager*
Apple Desktop Bus
AppleTalk Manager
Data Access Manager
Deferred Task Manager
Edition Manager**
Event Manager
Help Manager
Power Manager
PPC Toolbox**
SCSI Manager
Serial Driver
Sound Manager*With the Alias Manager, the use of file IDs is limited. The application engines won't notice if a user removes aliases of UNIX files.
**The Edition Manager, Event Manager, and PPC Toolbox are fully implemented, but they act the same way as the Macintosh Operating System with AppleTalk turned off.

AppleScript, an Elemental Technology — Why Using it is No Mystery

By Amanda Hixson

Go ahead, don’t use AppleScript! It’s just another scripting language—right?

Wrong!

AppleScript is possibly one of the most overlooked, least appreciated, and most important components of Apple’s current technical strategy.

”Whoa! Wait a minute, Holmes! How can you say that, especially after all the hype at the recent Worldwide Developers Conference about Amber, Apple’s new cross-platform, task-centered, compound-document software architecture? And what about Apple’s poignant appeals to adopt the Apple Open Collaboration Environment (AOCE), Newton, Bedrock, PlainTalk (speech recognition), PowerPC, and all that other ‘neat’ stuff?”

“Elementary, my dear Watson: AppleScript is the glue with which you will be able to (and, in some cases, must) bind together many existing and future Apple technologies—including Amber, PlainTalk, AOCE, and Bedrock—in a cohesive and consistent manner.”

Holmes is right. To control your applications with speech, to realize useful voice recognition, and to interact with Amber, you must implement key AppleScript-related technologies such as the Open Scripting Architecture (OSA), which defines a standard way for applications to interact with a scripting system.

To achieve OSA’s goal of creating a standard way for applications to interact with scripting systems, OSA uses Apple events, the System 7 interapplication communications (IAC) messaging language (which uses events as verbs and objects as nouns) to send messages across machines or networks by means of object-oriented or procedural applications. Introduced by Apple in May of 1991, IAC is currently supported by a growing number of developers including Microsoft (Excel 4.0), Claris (FileMaker® Pro 2.0), Quark (Express 3.2) and Deneba (Canvas). To standardize interaction between applications and any OSA-compliant scripting system, OSA also relies on a specific object model, or naming system, for objects in an application.

While working with developers, Apple realized that the potential for duplicate event and object names with different meanings would be high if events and objects weren’t tracked or registered. Consequently, Apple’s work with developers over the last few years has yielded several categories called *event suites.*

Event suites reflect standard ways to perform specific tasks by categorizing events and objects into related areas. The suites are managed by Apple, and they grow as new events and objects are submitted to and approved by Apple. Events and objects in existing suites are the result of tremendous cooperation and effort between Apple and the developer community.

The *Apple Event Registry: Standard Suites* document is a dictionary of currently defined standard words used in Apple event messaging. It contains suites for Required, Core, Text, QuickDraw, Finder, and Miscellaneous events and objects.

Constructs (also called *words*) in the standard suites include Apple events, Apple event object classes, primitive object classes, descriptor types, key forms, comparison operators, and constants. By supporting the Required and Core suites in an application, you turn it into one that can be scripted by AppleScript.

Additional registries include the Apple Event Word Services suite and The Apple Event Education suite. To submit events and objects for registry consideration, or to find out if the name for an event or object you want to use already exists, look it up in the *Apple Event Registry* or submit your proposed events and objects via AppleLink to REGISTRY.

Although OSA and some of its related components have been around since 1989, a limited number of you have taken advantage of them. If you had, you'd be ever so close to having scriptable applications and being able to quickly adapt to new technologies such as Amber (to which the importance of OSA and Apple events can't be overemphasized).

It's not too late to start implementing OSA and adding full AppleScript capabilities to your applications. Doing so will ensure your application's ability to communicate with any application or architecture supporting OSA, Apple events, and the object model. Some non-Apple products to which you will give yourself access after adopting AppleScript include Kaleida Labs' ScriptX, General Magic's Telescript, and even Microsoft's upcoming Object Linking and Embedding environment, version 2.0 (OLE 2.0). By using AppleScript and Amber, applications will work correctly with OLE 2.0.

By adopting AppleScript, you can take advantage of existing OSA-compliant third-party products such as Userland's Frontier and CE Software's QuicKeys.

When all of the previously mentioned opportunities are added together, the importance of AppleScript as a developer tool can’t be denied.

Don’t scold yourself because you thought AppleScript was just another scripting language. Maybe you missed it when Apple told you it was designed to deliver easy-to-use, system-level, application-consistent scripting through which you can add value to your existing applications and allow your users to tailor applications and their desktops to meet specific needs.

Possibly you also weren’t aware that you can also use AppleScript to help value-added resellers (VARs), in-house developers, consultants, and system integrators (SIs) modify client environments quickly and efficiently. Don’t fret—use the information in this article to help your development team make the decision to fully support AppleScript.

Who Should Use AppleScript

As described earlier, developers wishing to keep pace with Apple’s strategic planning should consider adding AppleScript support to their products. In addition to its critical role in long-term planning, however, AppleScript can play a key role in your application or computing environment today.

Commercial Developer Benefits. Commercial developers taking advantage of AppleScript’s object-oriented design and adherence to Apple’s event-messaging system can anticipate a number of benefits. First, AppleScript allows you to expand existing application capabilities because it provides universal, standard functionality. This eliminates the need to write application-specific scripting tools for every application.

Compiling scripts that are not (human) language- or dialect-specific is another benefit. Because compiled scripts are stored in a format called Universal AppleScript, users can read the script anywhere else in the world, regardless of the human language that each user speaks.

For example, Universal AppleScript allows a script to be moved from a machine running English system software to a machine localized for KanjiTalk; when viewed through a script editor, Universal AppleScript scripts appear in the script appropriate to that system: in this case, Kanji. (Comments, of course, won’t be translated, but AppleScript will display a script’s code in the user’s language.)

This AppleScript feature also enables programmers' scripts written in a C-like programming language to appear in a more familiar format, such as English, viewed by a user through a script editor.

As Apple moves to a more document-centered view of computing, scripting allows you to expand the features of your application without major modifications to core code. Events can be intercepted by scripts before reaching your application. Once intercepted, those events can be used to perform tasks, retrieve information such as that stored in shared libraries, and interact with other applications or other event-driven sources external to your application. If, for instance, you ship four products, AppleScript could be an ideal way for those products to all share common libraries in ways invisible to your users without the need for you to modify your core code.

Imagine that you sell a spreadsheet product that performs minimal database retrieval. Perhaps you've been hesitant to expand its capabilities to include databases that are external to your application because you don't want to have to learn how to use or maintain them. Using AppleScript, you could work with various developers to create access to specific external databases, thus eliminating the need for you to develop any specialized retrieval or content expertise for those database products.

With a technology as versatile as AppleScript, it's a safe bet that many commercial developers will create new software categories such as "smart scripts" (also known as *agents*) that flit around in the background doing specialized tasks triggered by particular events. Inevitably, someone will use AppleScript to create new categories of software not yet dreamed of.

Solution Provider Benefits. As mentioned earlier, solution providers such as VARs, consultants, in-house developers, and SIs are obvious beneficiaries of AppleScript's features, and they will derive additional benefit as more applications support AppleScript. Here are some areas in which solution providers will advance:

• You can create enterprise solutions using off-the-shelf applications combined with AppleScript. For example, a database application could be connected to a word processor and a spreadsheet and accounting package to automatically do payroll, create invoices, and even determine if users have the proper security to perform specific tasks.

• You can use scripts to automate tasks that should occur during off-peak hours, such as automatically connecting to on-line services and retrieving specific information, backing up network servers, checking for network problems (and even responding to them in some cases), performing customized software installation, and much more.

• Scripts can automate training for software applications, business procedures, school curriculum, or combinations of these and many other tasks.

• AppleScript can allow clients to use familiar products for new tasks. One of the largest financial drains on any large company involves familiarizing employees with new technologies, especially when those technologies are burdened with unfamiliar interfaces. Using AppleScript, solution providers can write scripts that tie new activities to familiar applications. For example, a client or user might use a common word processor to retrieve data from an SQL database by triggering a simple script using an action common to that word processor, such as a sequence of keypresses.

In complex environments, AppleScript can also be used to minimize programming efforts by eliminating redundant tasks and automating various processes like compiling code or building programs.

User Benefits. Although only the earliest of adopters from the user community will do much with AppleScript, others will become involved as more and more applications support AppleScript:

• Apple expects that most users will use AppleScript to automate tasks of various types. Some users will primarily automate repetitive tasks, others will use scripting to automate key, highly complex tasks that only occur infrequently.

• For some users, nothing is more fun than having a personalized desktop unlike anyone else’s. Toward that end, you can expect to see some intriguing uses of AppleScript to modify desktops and working environments. On machines used by more than one user, you might even see a log-on script tied to a security program that activates different desktop setup scripts based on user names or passwords.

How Do You Support AppleScript?

AppleScript is the continuation of Apple’s commitment to create the most flexible and innovative scripting environment possible. As mentioned earlier, key AppleScript building blocks include the OSA, IAC, and the object model (a methodology that allows script users to refer, in multiple ways, to objects—files, words, paragraphs, spreadsheet ranges, and so on—created by applications or scripts). Because the foundation on which AppleScript is built is highly flexible, there are currently three main ways for commercial developers to support AppleScript to allow solution providers and users to take advantage of AppleScript's capabilities:

• Making applications scriptable means giving your application the ability to understand Apple events sent by scripts. To be scriptable, an application should support the Required and Core suites and whatever other suites make sense for their particular application and should expose whatever functions inside itself that might be useful to other programs. Applications that can understand script-generated events are also controllable by scripts.

• Adding recordability allows your application to send events to itself. Being able to send and receive events lets your application send events that are related to specific user actions to the Apple Event Manager. AppleScript can then record those actions and store them in the form of a script. Scripts created using this method can be compiled, run, rerecorded, or modified by users with access to a script editor such as the simple one that comes with AppleScript.

To best implement recordability, you should first add scriptability to your application in order to bridge the gap between user interface code and your code. (The process of hiding the programmatic guts of an operation from the user is also called *factoring.*)

• Developers can also make their applications attachable, meaning that the script is hidden inside the document file; this eliminates the need to ensure that you have both the document and its script (in separate files). As described earlier, a security application might perform different actions, such as triggering a user-specific desktop configuration, based on passwords, security levels, or user names.

Attachable scripts are also referred to as embedded scripts. Embedded scripts are a data type and need not be created by AppleScript. A script created using another scripting language such as Userland's Frontier becomes an embedded data type when you add attachability to your applications. This is also true for low-level events, such as those created using macro languages, and even extends to include sounds, XCMDs, and other related structures.

Attachability lets you unify scripts and applications into complete environments, obviating the need to interact with remote script engines. Add embedding to your applications and you open them to customizability from within, connection to other applications, and even turning them into user agents.

A key to attachable scripts is the Open Scripting application program interface (API). It is also important that you understand the Macintosh Component Manager, as it is integral to embedding (and adds AppleScript to the list of technologies that, along with QuickTime, use the Component Manager) and acts as a general-purpose dynamic linker for AppleScript.

The form of AppleScript support you incorporate depends on the type of application you write and the circumstances under which it will be used. Full support of AppleScript isn't painless, but the potential advantages should anesthetize you long enough so that you can recover your initial investment, plus a great deal more.

More About the AppleScript Language

First and foremost, AppleScript is a programming language and, as such, has its own API that you need to support to add scriptability to your applications.

AppleScript's language is object-oriented and includes object classes and hierarchies. It accommodates single-parent inheritance from one script object to another script. People who write scripts (scriptors) define objects and their behaviors, and script objects can act as background agents, can be called as commands just like application commands, and can be contained in libraries. AppleScript also allows persistent data storage for scripts.

AppleScript is high-level programming language. It is easily understandable, and has syntax that includes constructs for looping, conditionals (if-then-else), setting variables, manipulating lists, and so forth.

AppleScript supports numerous data types including integers, reals, dates, times, text, lists, records and any application-specific formats. As mentioned earlier, AppleScript lets you work with multiple writing systems, such as Kanji and French, and AppleScript is also extensible in the following ways:

• Any application-generated objects or events that an application makes available to AppleScript extend the capabilities of AppleScript itself by expanding the basic AppleScript syntax.

• Linked routines written in any programming language extend AppleScript's capabilities in a way similar to the XCMD mechanism in HyperTalk. Unlike HyperTalk, AppleScript is available systemwide and is not restricted to influencing one application (as is HyperCard).

Scripting additions are actually event handlers or coercions (force-fed data) dynamically loaded by the AppleScript extension. Three types of addition resource files exist: executable 'OSAX' code needed by the addition;'aete' resources that expose the addition's commands and objects; and owned resources, such as dialog definitions, sounds, and so forth. Scripting additions are added to the System Extensions folder.

When compiled, scripts end up either ready to run or as stand-alone applications. You currently need to have the AppleScript Run-Time Kit to create AppleScript extensions that can be dropped into a user's System Folder to run scripts you write. (The AppleScript Run-Time Kit is available through APDA; see page 31 for ordering information.)

When the next version of Macintosh system software ships, AppleScript run-time extensions will automatically be installed as a player in a user's System Folder during installation. Therefore, if you don't intend to ship scripts before the next operating system reference release, you won't need to purchase the AppleScript Run-Time Kit.

AppleScript does come with a simple editor/recorder that allows you (and users) to create, change, or record scripts. Apple intends to provide more powerful scripting tools in the future, but there are developer opportunities here for building and editing your own editing tools.

Sample Scripts

At the end of this article are some sample scripts shown by Mark Minshull, AppleScript engineering manager, at the recent WWDC.

Notice in the Script 1, as shown in "Sample Scripts" on the facing page, that AppleScript is working across a network, using a "tell" statement to locate an application—Microsoft Excel—located on a Macintosh named Finance Computer located on a network zone named Building 1 to create a new chart document:

Script 2 uses "set" to modify the font size attribute of every word it finds in boldface to 36-point type.

Script 3 shows how a conditional (if-then-else) statement is used to check a category field to see if it contains the word *preferred*. If it does, then the script will change the price for a one-dollar product to 75 cents; if not, it sets the price to one dollar and 15 cents.

Script 4 sets a variable list of information (enclosed in braces) for an item named *Smith*. Once the list is set, a new variable *y* is assigned the value from the list corresponding to Smith's salary; thus, *y* now equals 55000. You probably noticed in the previous examples that AppleScript has some common commands such as "tell" and "set." Other AppleScript commands include "dialog," "display," "get," and "count."

As a developer, you can expose commands used by your application to AppleScript to expand the AppleScript vocabulary. You might need to add commands such as "cell," "row," "field," "morph," and "rotate" that are specific to your application in order for AppleScript to do everything your application permits.

At this point, we have only scratched the surface of AppleScript's features and capabilities, but don't despair—plenty of other information is available. You can find a list of resources on the previous page.

Future Directions

From this point on, you'll see AppleScript evolve in three directions:

• As mentioned earlier, AppleScript is a foundation for taking existing Apple technologies to new levels. It can be combined with QuickTime to create new forms of communication. Along with AOCE, AppleScript offers interesting new interactions across Macintosh environments and can be used to create special background agents that require authentication, store-and-forward services, or other AOCE capabilities. AppleScript will also act as a foundation for developing technologies such as voice recognition and Amber (into which you can expect to see AppleScript tightly integrated).

• AppleScript also provides the potential for encouraging the creation of new, powerful scripting tools such as enhanced editors. Expect also to see AppleScript integrated with development tools used with Amber and, possibly, Bedrock.

• Apple intends to do much in the area of cross-platform work. To do this, AppleScript will evolve from a Macintosh-only network product to accommodate much broader network interaction. Investigations are already underway to put AppleScript on other platforms, such as Microsoft Windows. Apple's PowerOpen, VITAL, and PowerPC projects are prime targets for making good use of AppleScript.

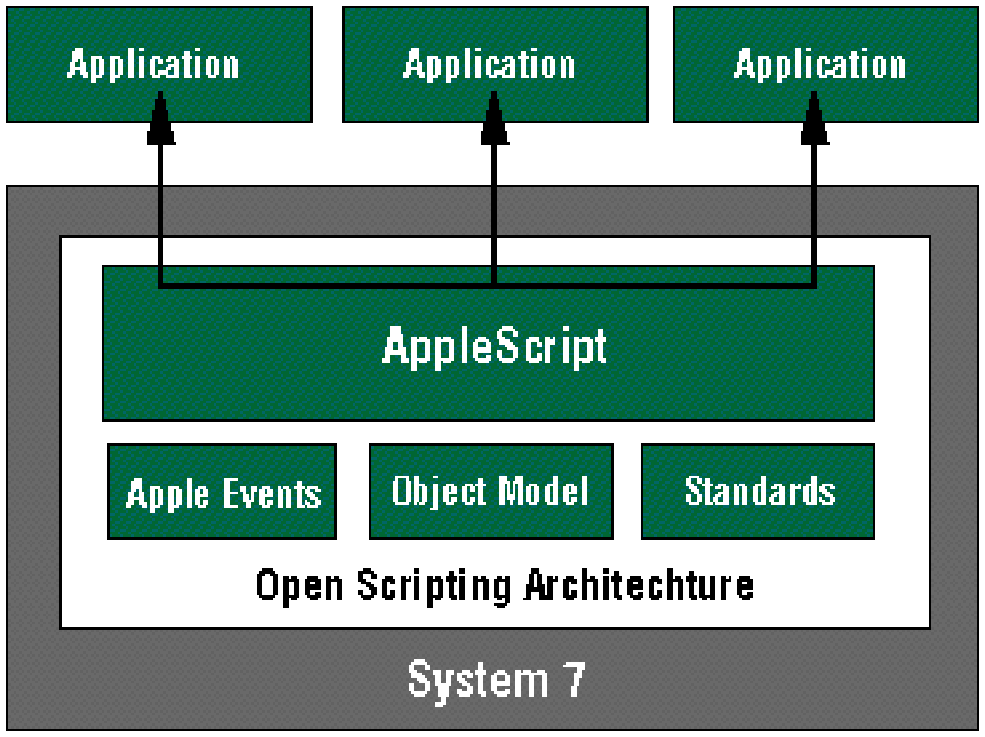
There isn't enough space to say much more than "What are you waiting for? Support AppleScript today!"

Amanda Hixson heads Instant Insight. She's a journalist and personal computer industry consultant.

****

How AppleScript Works

This model shows AppleScript's relationship to the system. AppleScript sits on top of the Open Scripting Architecture within the operating system along with Apple Events, the object model, and additional OSA standards. From this model, you can also see that AppleScript has easy access to applications.



Sample Scripts

Script 1

```
tell application "MS Excel" of Macintosh
"Finance Computer" of zone "Building 1"
create new chart
end tell
```

Script 2

```
set size of every word whose style is bold to 36
```

Script 3

```
if category = "preferred"
  then set price to retail * .75
  else set price to retail * 1.15
end if
```

Script 4

```
set Smith to {number: 15491, salary: 55000, manager: "Woo"}
set y to Smith's salary
```

****

Additional Resources

As is usually the case, there are many places for developers to turn for further information from Apple on AppleScript. The following section points you to some additional information sources.

Publications

• In the Apple Event Manager chapter of the forthcoming *Inside Macintosh: Interapplication Communication* (available from Addison-Wesley and APDA in July), you can read updated information about Apple events, the Core event suite, and the object model. In the interim, you can order the current Developer CD for an electronic version of the current version of *Inside Macintosh*.

• A self-paced Apple Events/AppleScript tutorial by Apple Developer University is also available from APDA (part number R0224LL/A) at a cost of $150.00 in the United States). See page 29 for APDA ordering information.

• You might want to consider APDA’s special bundle offer (good through September 30, 1993) that includes the above tutorial and the entire AppleScript Developers Toolkit version 1.0 (part number B1282LL/A) for $275.00 in the United States—a savings of $74.00 off the individual prices.

Tools and Sample Code

• Available through APDA, AppleLink and/or the Developer CD Series

• the AppleScript Developers Toolkit, which includes tools, sample code, the *Language Guide*, and *Getting Started With AppleScript.*

• the Apple Event Registry: Standard Suites collection

• the Apple Events Word Services Suite

• the Apple Event Education Suite

Training

• Apple Developer University offers both a class and a self-study guide on Apple events/AppleScript. For information on the class, contact Developer University (in the United States) at (408) 974-6215 or send electronic mail to them at these addresses: DEVUNIV (AppleLink) or devunit@applelink.apple.com (Internet).

On-line Resources

• Become part of the AppleLink electronic information service. (A sign-up package is available from APDA.)

• Submit your proposed events and objects via AppleLink to REGISTRY.

• For seeding information, use AppleLink and send your request to APPLESCRIPT.

• Also see the AppleScript Talk Bulletin Board on AppleLink (path—Developer Support:AppleScript Talk).

The Games Marketing Game
Playing to Win in the Entertainment Market

By Kris Newby

If you've dismissed the Macintosh entertainment software market as a trivial pursuit, take another look. With Macintosh sales to home users on the rise, the demand for games and fun, educational "edutainment" software is increasing dramatically. "All the easier-to-use, low-priced, mass-market Macintosh computers—in particular the Performa models—are really making an impact," says Bruce Fredericks, senior marketing manager of education and entertainment products at Brøderbund.

Successful developers in the Macintosh games market say there's tremendous upside potential in it, and compared to DOS and cartridge games markets, the Macintosh playing field is relatively uncrowded. (For more information, see "The Size of the Macintosh Games Market" on page 24.)

In the games business, success depends on making the most of a nine- to twelve-month product shelf life with precisely timed PR blasts and well-oiled distribution channels.

In this article, the marketing experts at six successful Macintosh games companies—Interplay, Maxis, Atreid Concept, Image Smith, Graphics Simulations, and Brøderbund—share their stories on how they got started and what they think it takes to win in the games market. (For a brief profile of each company, see "Developer Success Stories" on page 27.)

Your Game Plan

Unlike many other software markets, in the games business you can choose how involved you'd like to get with a product's marketing effort. If you're a novice, you can have a publisher handle everything, or you can share marketing responsibilities with a copublisher. If you already have established retail distribution channels, you'll probably want to manage your own marketing program. But as with any business, the more up-front money you invest and the more risk you assume, the greater your potential return.

Using a Publisher. When you use a publisher to market your product, your share of the profit will be smaller, but you also minimize the risk, time, and money it takes to break into the market. It also frees you to spend the next year doing what you probably enjoy the most—developing products.

Also, based on its experience, a publisher can help advise you about whether it thinks you have a hit product. (But do keep in mind that it's just one opinion, and if a publisher turns you away, don't let this, alone, discourage you.)

After spending many years in the computer-aided graphics and packaging design business, Dominique Claessens, president of Image Smith, recently went through the school of hard knocks in introducing his first product, Yearn 2 Learn—Peanuts. "Unless you have the sell-through rates, unless you have more than one game product, unless you have money for all of the promotional activities, be prepared to have a painful experience publishing your own products," says Claessens.

So, what kind of product royalty can you expect if you market your product through a publisher? Depending on your track record and negotiating skills, you'll probably receive a 7–15 percent royalty based on your product's wholesale revenues.

For example, if your game lists at $60, sells for a street price of $40, and wholesales to a retailer for $22, you'll receive about $2 for every product that sells. In this scenario, you'd be responsible for providing the publisher with finished product code, and the publisher would do everything else—packaging, PR, marketing, and distribution.

Copublishing With an "Affiliate Label." If your long-range goal is to be a big player in the entertainment market, but you don't have the initial capital that it takes to properly introduce the first product, consider copublishing your product with a larger "affiliate label" publisher.

Generally, in copublishing relationships, the games designer finances and manages the manufacturing and user marketing of a new product, while the affiliate publisher assumes channel marketing responsibilities. The table "Typical Copublishing Marketing Responsibilities" summarizes a typical division of marketing responsibilities in a copublishing relationship.

"Working with an affiliate label partner has real advantages in terms of getting distribution. You also benefit from having a top-notch sales force without the overhead that it involves. But as your company grows bigger and your products gain market presence, it makes more sense to begin moving away from a copublishing relationship," says Robin Harper, vice president of marketing at Maxis. That company comarketed its first product, SimCity, with Brøderbund in 1989.

According to Harper, the downside of copublishing is that it may be difficult to establish your own brand identity when you decide to break away from the affiliate partner. Also, as an affiliate label you have very little influence over your partner's sales force, so if a partner chooses to ignore your product over another in the line, you're out of luck.

So, what's your slice of the pie if you go this route? Copublishing arrangements vary widely, but typically you'll be responsible for up-front marketing and manufacturing costs. These costs are typically $10,000–$25,000 for a PR campaign, $15,000–$25,000 for a box design (from concept to color separations), and about $60,000 for an inventory of boxes, disks, labels, and other such items.

In return, depending on your arrangement you could receive 50 percent of wholesale revenues from the parent publisher. In other words, if your product wholesales for $22, you would receive about $11 per unit sold.

A common alternative to sharing wholesale revenues is to arrange with an affiliate label distributor to sell the affiliate the product at a price below usual wholesale (say, 10 percent lower).

Publishing on Your Own. If you're up to the challenge of creating your own entertainment software empire, read on. You'll benefit from the experience of six developers who beat the odds and
are currently selling some of the biggest entertainment titles in the business.

Regardless of whether you publish a product solo or with a partner, you should start planning a marketing strategy at least six months before a product ships. Even if you're not marketing the product yourself, it helps to have a rough marketing plan in hand when you approach prospective publishers; it makes them feel you've done your homework, and gives them ammunition to help convince their management team to publish your product.

In writing your business plan, make sure you consider these unique aspects of the entertainment market:

• *Target market.* Keep in mind that in the entertainment market, you may have to simultaneously target marketing messages to two audiences: a child and a parent. In your plan, profile the age range, sex, and demographics of target users. If your game is designed for children under the age of ten, the parent will make the purchasing decision; so you may want to emphasize the educational aspects of your product, and spell out the demographics of the parent group you want to attract.

• *Creative statement.* In the games market, differentiation is crucial. You should clearly define what makes your game unique in the market and what the important messages to deliver to the channel and your target market are.

• *Channel strategy.* Your best bet is to sell in both mail order and retail channels, but you may have to establish a sales history in mail order before a retailer will pick up your product. (If you plan to sell to educational establishments, see "The A-B-Cs of the Edutainment Market," later in this article, for tips on how to boost your sales to educators.)

• *Introduction plan.* Work up a rough timeline for your product introduction. How soon will you be able to offer prerelease copies to industry reviewers and opinion leaders? Is there a major consumer trade show during the time frame when your product will be ready to launch? Do you have ideas about how to generate excitement around your product? Does the product demo well to crowds? Can you create a limited-play demo product version for widespread distribution?

• *Packaging.* All the developers we interviewed said that aside from having a great product, packaging was the single most important element of a marketing plan. (See "Design Eye-Catching Packaging," later in this article, for advice on this topic.)

• *Pricing.* Pricing in the Macintosh games market is fairly narrow, with retail list prices clustered between $30 and $60. In choosing a price for your product, research the retail and street prices of competitive games.

Based on the competition, your product's technology content, and what you think your target audience will pay, decide if your product should be positioned at the high or low end. Usually you'll see simple action games on the low end of the pricing spectrum, and products with more technological and educational content on the high end.

• *Cross-platform development plans.* Many developers we interviewed initially create their products on the Macintosh and then port them to other platforms. The Macintosh has a powerful assortment of graphic and sound tools available to help create games.

In talking about why he chose to develop on the Macintosh platform first, Image Smith's Claessens says, "We will probably always develop first on the Mac—that's where we do all of our authoring. But I think that anyone who's creating software today has to do
it cross-platform to grow their business."

All but one of the developers we spoke with develop for multiple platforms: They see it as a way of leveraging product development costs and minimizing risk.

Steps for Marketing Success

Games marketers we interviewed each gave us a formula for successfully marketing entertainment software; a consistent set of "steps for success" emerged. The following information will help you benefit from their experience in orchestrating word-of-mouth PR campaigns, designing eye-catching packaging, and maximizing the impact of advertising dollars.

Introduce a Great Product. Though it may seem too obvious to mention, almost every entertainment developer we talked to said that having a unique and fun-to-play product was the most important factor of its marketing success.

Jeff Morgan, president of Graphics Simulations, said he knew that the company's flight simulator, Hellcats Over the Pacific, had the "right stuff" when Joe Montana, the football quarterback, called him to say, "My wife hates you guys." (I guess Jennifer Montana wanted to unplug the computer so Joe could help pack for their move from San Francisco to Kansas City.)

"There are two aspects of a game that make it great," says Nicolas Gaume, president of Atreid Concept, a games developer in France. "The most important thing is the game play—the pleasure you get out of a game because it's different and new or better than the competitor's product. The other is the game's overall visual effect, because that's the first thing that attracts users to it."

Product testing is a key element in creating a great game. "To make your product a polished piece of art, it's very important for a publisher and developer to make sure that the game has been tested—not only by the original development team, but by target users and reviewers," says Atreid's Gaume.

Image Smith even goes as far as to test their edutainment products' human interface by performing eye-tracking tests. In these tests, users wear goggles that allow testers to trace how a user's line of vision travels across a screen, so that the effectiveness of cursor designs, icons, and animations can be determined.

Orchestrate a Word-of-Mouth PR Campaign. The unique aspect of marketing entertainment software, in contrast to such applications as business productivity software, is the almost-out-of-proportion importance of generating word-of-mouth PR.

Though using this marketing technique is less expensive than having to rely on advertising or direct mail campaigns, harnessing its power is much more elusive. It requires more staffing, persistence, and creativity. Much like the movie business, this market is driven by "hits." Success depends on quickly implementing a PR campaign to make the most of a nine- to twelve-month product life and building a reputation of being a hit product.

Orchestrating the timing of your product introduction is critical. A strategy that works for many developers is introducing a product at a big consumer trade show (such as Macworld Expo or the Consumer Electronics Show). At least four months before the trade show, send out demo products to reviewers. This will help ensure
that product interest is on the rise prior to the introduction. At the conference, stage a crowd-pleasing demo and sell products directly from the floor.

Jeff Morgan of Graphics Simulations says that the word-of-mouth PR generated at Macworld Expo launched his company. "We set up big monitors so that as people walked by, they could see our flight simulator in action. As a result, we sold a large number of products at the show. People went out and told all of their friends, and before we knew it, sales took off," he says.

Here are some other ideas for initiating a word-of-mouth PR campaign:

• *Send a product demo and information to Macintosh user groups by mail or electronic mail.* "There are literally hundreds of good user groups out there that will get word-of-mouth going," says Bruce Fredericks of Brøderbund. (For information about how to participate in Apple's user group contact efforts, call the Developer Support Center at (408) 974-4897, or send an AppleLink message to USER.GROUPS.)

• *Create a "scene" at trade shows with a demo or competition.* It's best to avoid feature- and benefit-oriented product demos at trade shows. Instead, engage trade show attendees with theatrical and interactive events. For example, Brøderbund, creator of Where in the World Is Carmen Sandiego, recently dressed up booth workers in fedoras and trench coats, and picked volunteers from the audience to search for Carmen Sandiego.

• *Get publication and electronic mail games reviewers to write about your product.* Good product reviews can sell a lot of units and stimulate sales generated by word-of-mouth PR. (For more information, see "Getting Product Reviews" in the October 1991 issue of *Apple Direct*, available on the June 1993 Developer
CD entitled *ROM the World in 80 Nanoseconds.*)

• *Upload a product demo to Macintosh game forums on major on-line services.* AppleLink, GEnie, America Online, and CompuServe all have games discussion boards where good, informal reviews from games enthusiasts can help spread the word about your product.

• *Design a self-running demo for retail display systems.* If you have a visually engaging game, you may convince retailers to use it as standard demo software on their CPUs. This results in "free" advertising for your product.

• *Visit local retailers to generate enthusiasm.* In addition to sending product information and a demo disk to retailers, try arranging a "vendor day" with local retailers. Draw a crowd by offering product discounts, autographing products, and organizing competitions.

• *Find Apple employees and industry spokespersons to help "evangelize" your product.* To recruit more proponents for your product, liberally hand out demo disks when you travel to trade
shows and visit Apple offices. Who knows, you may even convince someone at Apple to demonstrate your product in a keynote presentation or at a press event.

Design Eye-Catching Packaging. What you ship your product in is as important as the product itself. The box design is the primary vehicle through which you communicate with prospective buyers. And packaging will become even more critical as more Macintosh software makes its way onto crowded retail shelves.

Before you start designing a package, take a field trip to your nearest software store. With eyes closed, walk to the center of the games aisle. Open your eyes long enough to take a quick "snapshot" of the shelf. Then close your eyes and analyze which packages made the biggest split-second impression on you.

"The box is a critical tool for us," says Robin Harper of Maxis. "If you can't get someone to pick up a box in the first two or three seconds, you've lost a sale. Our goal is to get a potential buyer to turn the box over so we can clinch the sale."

Designing a uniquely shaped box can be an effective means of attracting buyers. (However, it's important to be keenly aware of the potential difficulties that unusually shaped packages present to retailers; they must find ways to shelve such packages, and in doing so may be forced to put your box in a less than optimum place.)

Brøderbund's Bruce Fredericks was surprised at how well the Prince of Persia game's new trapezoid-shaped box rejuvenated sales of this older offering. "Before we released the Mac version of Prince of Persia, we spent a healthy sum of money repackaging the product in an odd-shaped box. It was incredible how sales came to life! We knew it was a result of the package because it was an older product, and we really weren't putting more marketing dollars into any other avenues. Retailers also like it because they can stack it well, and it doesn't violate the
overall perimeters of a typical box form," says Fredericks.

Another packaging success story comes from Image Smith's Dominique Claessens, whose previous company was best known for the package design of Heineken beer. To differentiate his Peanuts product, he designed a doghouse-shaped package. He claims it's been a magnet in capturing impulse buyers, young and old.

If you're marketing several products, Phil Adam, vice president of marketing at Interplay, recommends combining compelling graphics with a product line "look" that customers can learn to recognize. (For a more comprehensive discussion of product packaging, see "The Ten Commandments of Product Packaging" in the September 1992 issue of *Apple Direct.*)

Focus Ads on Your Company Image. Most game developers agree that advertising is an expensive way to generate sales; if you don't choose your advertising targets wisely, you'll throw your money away. Here's some advice for getting the most out of your games advertising dollars:

• *Use ads to build a company image.* Since advertising is a long-term marketing tool and many games only have a nine- to twelve-month shelf life, consider using ads to build a company image rather than to highlight a single product. Work to create a company brand identity that buyers can count on to consistently deliver quality games.

• *Advertise in home buyer– focused publications.* A single-page ad in *Macworld* or *MacUser*
can set you back as much as $25,000. Game developers say that these publications' audiences may be too broad to get the response you need. Instead, they recommend that if you advertise, do so in publications focused on home buyers, such as *MacHome Journal, MacComputing, Computer Gaming World,* and *Kids Computing.* You'll be able to buy equivalent ad space in these publications for a quarter of the price.

If you sell edutainment products, investigate putting ads in publications such as *Parenting, Family Fun,* and *Parents Magazine.*

• *Buy co-op ads in direct mail catalogs.* In catalogs such as those offered by MacWarehouse and MacConnection, you can buy inexpensive ad space that reaches thousands of qualified buyers. A quarter-page ad costs about $2,500. (If you're trying to get picked up by a catalog, one good thing to know is that catalogs often track customer requests for a new product. So, if you're going to make a sales call on one of these mail-order companies, get all your friends and relatives to request your product the month before you contact a catalog's buyer.)

• *Use repetition to maximize effectiveness.* Studies show that advertising is most effective when run over a long period of time. It's better to run an ad in an inexpensive spot for six months than to spend the same amount of money on a one-time ad to appear on an inside cover.

Winning the Shelf Wars

Right now, 60–80 percent of all entertainment software sells through direct mail. But as the Macintosh moves into consumer venues such as Circuit City, Walmart, and Sears, getting shelf space becomes increasingly important. Game developers say that getting into retail channels is the biggest challenge to succeeding in the games business today.

Image Smith's Vice President of Marketing Jim Myrick theorizes on why it's so hard to get Macintosh games on retail shelves: "During the recession, a lot of distributors ended up carrying excess inventory and being burned by products that were hot sellers last week and dogs this week. And retailers feel that if mail-order businesses like MacWarehouse, MacZone, and MacConnection are going to sell most of the Macintosh games software, why should they allocate precious retail space to it?"

So what does it take to get shelf space? Having a market track record helps. If you can show a retailer that you've been successful in another market, on another platform, or through mail order, it'll be easier to get your foot in the door.

The authors of the *Software Channel Sales Guidebook* (published by Relaunch) recommend building a sales history with a mail order catalog, local distributor, or both before approaching the big retail distributors such as Merisel and Ingram Micro—that way it'll be easier to get their attention and negotiate a better deal. (For more details about what it takes to approach a national distributor, see "Understanding Distribution Realities" in the October 1992 issue of *Apple Direct.*)

Second, retailers like to work with software companies that have more than one product. It signals that you're not a flash-in-the-pan company, and that retailers will be able to minimize their inventory risk by "stock balancing" your products. (Stock balancing is the practice of retailers returning your slow-selling products and swapping them for your hot sellers to increase their revenues.) If you're just starting out, you'll increase your chances of having product sold through major retailers if you team up with a copublisher that can provide stock balancing.

And last but not least, start getting your company and products known in the channel by networking with retailers, distributors, and reviewers.

The A-B-Cs of the Edutainment Market

With the Macintosh education software market growing at an incredible rate—64 percent from 1991 to 1992—it's a good time to consider developing fun, educational "edutainment" software. (Edutainment products both entertain and educate users.) "Clearly there's a real education movement out there—I could sense it at the Games Developer Conference this year," says Bruce Fredericks of Brøderbund.

If you're introducing an edutainment product, developers say it takes considerably longer to get word-of-mouth PR going among educational institution buyers than home buyers. This is due to the slow-moving purchasing structure of schools. Here are some tips for getting a boost in this market:

• *License and use shelf-appeal characters in your products.* Turn on the Saturday morning cartoons to find characters that could help
sell your games to children and reduce the amount of money you need to spend on marketing programs. Make sure you license the rights to use these characters if you build them into your products.

• *Build multiple activities into your product.* Image Smith's Yearn 2 Learn product is a good example of how edutainment products with multiple activities allow you to sell to a wider age bracket of children and increase the perceived value
of the product in parents' eyes. In this product, a child can choose from subjects that include reading, three levels of math, two geography puzzles, a ten-page coloring book, and four Flying Ace games.

• *Get feedback from educators during development.* Educator feedback on your product's design is essential if you intend to market to both consumers and educational institutions.

• *Target word-of-mouth activities to educators and institutional buyers.* Try to get reviews and coverage in education trade journals, and demonstrate your product at education-specific computer trade shows.

Reaching Global Markets

Most U.S. developers have found Europe, Australia, and Japan to be the hottest international games markets. For example, Yearn 2 Learn is pulling a Japanese sales volume equal to U.S. sales, in spite of the fact that a Japanese version hasn't been released yet. And
Hellcats Over the Pacific from Graphic Simulations has had similar success in Japan.

With sales volumes like these, it's hard to ignore markets outside the United States. Experienced developers warn that if you're planning to market internationally, be prepared to sell your product differently in each country. They recommend finding a strong local partner to help you localize and market your product. (For more information about how to succeed in international markets, see "International Success Stories" in the August 1992 issue of *Apple Direct.*)

Games developers we spoke to said that outside the United States the shelf life for games tends to be shorter than the usual nine to twelve months, and there’s considerably less retail shelf space available. This makes the word-of-mouth hype that you generate all the more important.

In Europe, you’re going after a more diverse, fragmented market, and there’s a smaller middle class and less per-capita income than in the United States. If you’re an outsider to European markets, it’s harder to choose the best publications to advertise in—especially since there are 25 to 30 major computer publications, versus the four or five major ones in the United States.

“One very good marketing tool that we have in Europe is the ‘cover disk,’” says Gaume of Atreid Concept. “Most games magazines are mailed with a floppy disk on the cover, where we can include demos.”

Because of the smaller international window of opportunity, you’ll probably make more money if you release localized versions of your game simultaneously or soon after the English version of your game is introduced. A carefully localized product with a price within 20 percent of the U.S. price will also help you reduce gray market sales and software piracy in other markets.

And finally, remember to think globally when you design your software and package. Have a “native” review your localized product for local cultural sensitivities to words, animals, and symbols.

Jumping on the CD Fast Track

As more Apple CPUs ship with built-in CD-ROM drives and the overall installed base of drives increases, you may want to consider developing games specifically for this media. Apple forecasts it will sell 1.5 million CD-ROM drives in 1993. And Ian Diery, Apple’s executive vice president of worldwide sales and marketing, estimates that there will be more than four million Macintosh CD users in 1994. “Our goal is to become the predominant CD-ROM platform by the middle of next year,” says Diery.

Interplay’s business has demonstrated the potential of the CD market; its best-selling Macintosh product is the CD version of BattleChess. “CD users don’t really have that many good game products to choose from right now,” says Interplay’s Vice President of Marketing Phil Adam. “We try to utilize CD technology to its fullest, not just port over a floppy version of a game. Macintosh users with CDs right now are sophisticated buyers, and they appreciate what you can do with the technology.”

Interplay's upcoming adventure game, The Fabulous Fuzz Box, lets users experience state-of-the-art sound and graphics as they search through time for John Lennon's famous "Fuzz Box" amplifier.

One advantage of putting games onto CD-ROM is that, for the developer, the cost of the media is lower than the cost of floppies. Also, you can put a lot more code and graphics on CDs than on floppies.

What's more, industry studies show that CD-ROM owners spend more money, overall, on software. According to the *Boston Globe,* typical parents with home computers will spend upward of $250 on software for their children during their early school years. Parents
who also buy a CD-ROM player, on the other hand, usually spend up to $695 for educational software.

Another reason to sell CD games is that you'll lose fewer profits to software piracy, since it's impractical (and in some cases virtually impossible) for users to copy larger CD-based games onto floppy disks or hard disks.

The Game That Never Ends

As any avid game player knows, a great game never ends—after slaying the dragon, eating the magic strawberry, and melting the evil space aliens, your "reward" is to get to play the game on a higher, more difficult level.

You'll probably find that this is also true with your entertainment software business. Once you get past the initial hurdles in starting your business, you'll find that a new set of challenges awaits. Just remind yourself that if the game were easy, it wouldn't be fun.

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

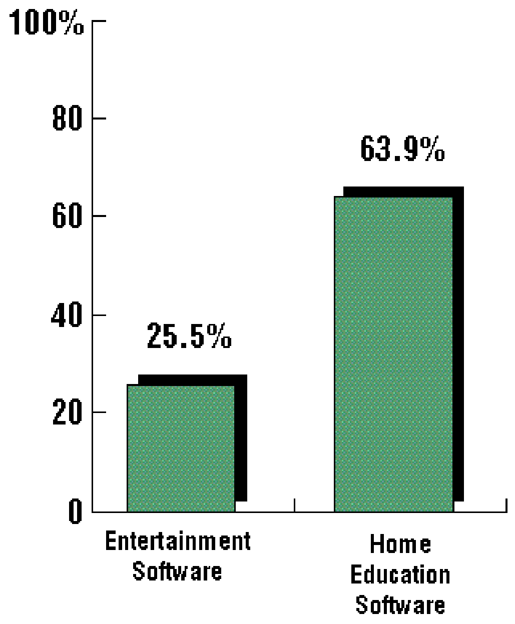
****

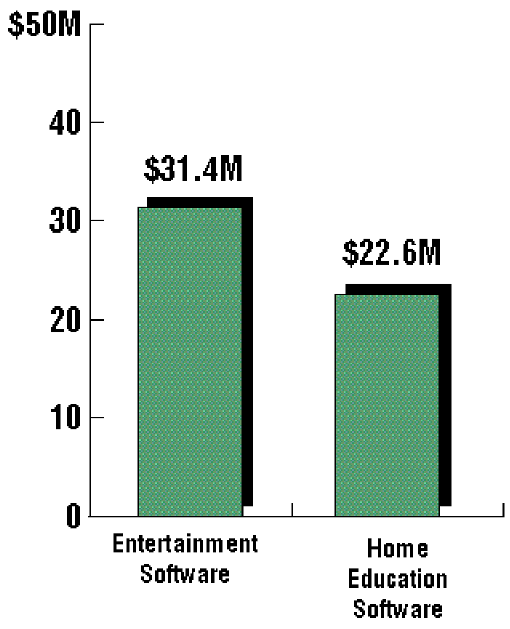
The Size of the Macintosh Games Market

At the start of 1993 in the U.S. home computer market, Apple sold more computers than any other company, according to InfoCorp. The availability of low-cost Macintosh Performa and color Macintosh LC computers has helped fuel this market segment's growth and created demand for more education and entertainment software. To get an idea about the size and growth rate of the

Macintosh entertainment and home software market, see the two bar charts below.

Macintosh Software:
North American Growth Rate
in Dollar Sales
Fiscal Year 1992 Over 1991





Macintosh Software:
Total North American Retail Sales
Fiscal Year 1992

Typical Copublishing
Marketing Responsibilities

Your Responsibilities	Affiliate Partner/ Distributor Responsibilities
Focus on user pull	*Focuses on channel push*
Design and manufacture product and packaging	Ships affiliate products through an established retail/distributor network
Solicit product reviews Post on-line demos and info	Creates demand through brand recognition
Generates sales through sales force prospecting and vendor days	Send out press releases and user group mailings

Create and place user advertising	Demos product at major trade shows, sometimes provides booth space
Produce retail “sell sheets” and demo disks	

****

Developer Success Stories

Here is a brief profile of each company we interviewed for this article.

Graphics Simulations. This company was started nine years ago as a small firm that wrote C compilers and contract software in Dallas, Texas. Out of boredom, Jeff Morgan and Trey Smith assisted Eric Parker in writing Hellcats Over the Pacific. After deciding to publish the product on their own, they introduced Hellcats at the January 1992 Macworld, and before they knew it, word of mouth took over and sales took off. Graphic Simulations is a four-person, Mac-only company, and it offers what many users consider to be the best flight simulator on the Macintosh platform. “We’ve had people say they’ve gotten airsick playing our game too long,” says Morgan.

Interplay. This firm’s best-selling entertainment titles are BattleChess CD and Out of This World. Interplay’s founder, Brian Fargo, founded his own company in 1983 and initially published products through Electronic Arts and Activision. Today this company, based in Irvine, California, is an established games developer and publisher with revenues of more than $20 million and approximately 100 employees. It markets games for the DOS, Nintendo, and Gameboy platforms, as well as for the Macintosh. Six months ago it launched a new division, MacPlay, to focus development efforts on the promising Macintosh market.

Maxis. Maxis shipped its first product, SimCity–The City Simulator, with affiliate partner Brøderbund in 1989. This product continues to defy the “laws” of entertainment software by showing strong sales in its fourth year on the market. About six months ago this 85-person company in Orinda, California, started publishing under its own label, with follow-up products that include SimEarth–The Living Planet , and SimAnt–The Electronic
Ant Colony.

Image Smith. This developer, located in Torrance, California, is a new player in the edutainment software market. It released its first product, Yearn 2 Learn–Snoopy, at the January 1993 Macworld Expo. Initial sales have been impressive. Dominique Claessens, the president and cofounder of Image Smith, formed the company after selling the computer operations part of his computer graphics and packaging business. The other cofounders, who have extensive experience in multimedia title development, include Jim Myrick, previously from Pixel Ink Consultants, Jake Myrick, a recognized Macintosh graphic design artist, and Noriko Kamei, an expert on multimedia title creation and localization.

Atreid Concept. This games developer and publisher employs approximately 30 people and is located in Bordeaux, France. The hallmark of Atreid games is graphics, sound, and entertainment value. Its best-selling product, Tinies, is doing well in Europe and the United States. In Europe, Atreid markets products through Kalisto, its publishing label, and in the United States it currently distributes products through Inline Design.

Brøderbund. This big name in educational software publishing is a $75 million company founded by brothers Doug and Gary Carlston. The company got its start in 1979 while Doug Carlston was practicing law in Maine, where he used his spare time to dabble in writing software. “I’d gotten a Radio Shack computer and started programming again. I ended up with three publishers selling this really simple stuff I’d written,” says Carlston. Soon the software was generating more income than the law office, and the rest is history. Brøderbund is headquartered in Novato, California, and its current top sellers are Where in the World Is Carmen Sandiego? and Prince of Persia.

****

For More Information

To get more information about starting out in the Macintosh games business, Craig Fryar (Apple’s games evangelist) and developer interviewees recommend two publications. First, Relaunch’s industry bible Software Channel Sales Guidebook is a street-savvy, sometimes irreverent guide that offers advice on getting your products into the hands of users. (It can be ordered by phone: 800-875-9099 or 510-528-9099.) Second, for great practical advice on writing and marketing your games, get a copy of Apple’s Macintosh Game Developer’s

Handbook. It’s available on the 1993 WWDC CD. Both of these publications include contact names and numbers at major software distributors and retailers.

You may also want to check out the Game Development Discussion on the AppleLink network (path—Developer Support:Developer Talk:Game Development Discussion).

It Shipped!

Through the It Shipped! program, you can announce new and revised third-party products in *Apple Directions.* It Shipped! listings are also made available on the 3rd Party Connection AppleLink bulletin board. You can obtain an It Shipped! application by downloading it from the AppleLink network (AppleLink path—Developer Support:Developer Services:Apple Information Resources:Developer Program Information:It Shipped! @ Program).

Once you've completed the application, send it to Engineering Support, Apple Computer, Inc., 20525 Mariani Ave., M/S 42-ES, Cupertino, CA 95014, Attn: It Shipped! Program. Or send it by AppleLink to IT.SHIPPED.

The following products shipped in June 1993:

Brio Technology, Inc.	• DataEdit 1.0
Graham Software Company	• Talking Keys Pro 1.0
Image Club Graphics, Inc.	• Art & Type Vendor CD-ROM System 2.0 • newFaces Spring 1993 Type-face Collection 1.0 • PhotoGear CD ROM Volume 1 1.0
Johnathon Freeman Device Server 1.0	• ParaLink Plus–Virtual Parallel Technologies • SeriLink–Virtual Serial Device
Server 1.0	
Palomar Software, Inc.	• On the Road 1.1
Proteus Technology	• Quota 3.0
Skills Bank Corporation	• Cornerstone Language 1.05b
The Multimedia Library SOUND Royalty Free, CD-ROM	• Ethnic Music For Multimedia - Series, Vol. 2 • The Holy Land–IMAGE Royalty Free CD-ROM Series, Vol. 9

Hot Products From APDA

Special Introductory Offer!
AppleScript Software
Development Kit Bundle

Build AppleScript into your applications to support custom solutions for your users. Get in on the scripting revolution with this special offer.

Order the AppleScript Software Development Kit and Tutorial Bundle before September 30, 1993, and get 20% off!

AppleScript is Apple’s powerful new scripting system that works across applications and networks to deliver automation, customization and application integration capabilities for the Macintosh.

AppleScript will make custom application development faster and easier because commercial applications can be used as components with AppleScript acting as the “glue” between them.

Look what AppleScript offers you when you build it into your application:

- Increases the value of your products - Apple is making scripting an integral part of the Macintosh computing environment
- Positions you to take advantage of emerging technologies like speech recognition
- Lets you expand into new markets - Apple is opening up new business markets that who will demand AppleScript-compatible products
- Allows for customization - AppleScript will allow your products to accept functional add-ons
- Reduces time to market by allowing your applications to be combined with other packages
- Creates a cross-platform scripting foundation.

This special bundle contains both the AppleScript Developer’s Toolkit and the Apple events/AppleScript Programming Tutorial. The AppleScript Developer’s Toolkit is designed to help you build AppleScript support into your applications. It contains a Macintosh disk, one CD-ROM disk, AppleScript Language Guide, and Getting Started with AppleScript. The Apple events/AppleScript Programming Tutorial is a self-paced disk and workbook tutorial that teaches you the basics of the AppleScript architecture.

B1282LL/A   Special price $275.00 (U.S.)

More Special Offers

New Inside Macintosh —20% off each book!

- *Inside Macintosh Files:* P0025LL/A
  Regular Price $29.95 ***Special price $23.95 (U.S.)***
- *Inside Macintosh Overview:* P0026LL/A
  Regular Price $22.95 ***Special price $18.35 (U.S.)***
- *Inside Macintosh Macintosh Toolbox Essentials:* P0029LL/A
  Regular Price $34.95 ***Special price $27.95 (U.S.)***
- *Inside Macintosh Processes:* P0033LL/A
  Regular Price $22.95 ***Special price $18.35 (U.S.)***
- *Inside Macintosh Memory:* P0037LL/A
  Regular Price $24.95 ***Special price $19.95 (U.S.)***
- *Inside Macintosh QuickTime:* P0038LL/A
  Regular Price $29.95 ***Special price $23.95 (U.S.)***
- *Inside Macintosh Text:* P0039LL/A
  Regular Price $39.95 ***Special price $31.95 (U.S.)***

Macintosh Human Interface Guidelines Bundle - 28% off!

Special bundle includes the new version of *Human Interface Guidelines* plus the new *Making it Macintosh Human Interface Guidelines* CD-ROM featuring more than 100 animated examples of interface design practices.

P0043LL/A   Regular price $69.90 ***Special Price $49.95 (U.S.)***

Special savings on Developer University Self-Paced Training Products!

Save up to $395 on DU’s self-paced training courses. Until September 30, 1993, you can order either Macintosh Programming Fundamentals or the new Intermediate Macintosh Application Programming course for only $395! Or, buy them both at the special bundle price of $695!

Macintosh Programming Fundamentals
Build highly-functional Macintosh applications through mastery of fundamental Macintosh ROM routines and the application programming interface.
M0997LL/B Regular price $595.00 (U.S.) S
Special price $395.00 (U.S.)

New! Intermediate
Macintosh Application Programming
Extend you knowledge and Macintosh programming skills beyond the basics. This course teaches you how to build on the skills you learned in the “Macintosh Programming Fundamentals” course. Write code that extends the functionality of a single-finder graphics editor to include QuickTime movies, Publish and Subscribe, Cut, Copy, Paste, TextEdit, TrueType Fonts, required Apple events, MultiFinder, and more.
R0438LL/A Regular price $495.00 (U.S.)
Special price $395.00 (U.S.)

MPF/IMAP Bundle
Get Both and Save $395!
Contains both the Intermediate Macintosh Application Programming course and the Macintosh Programming Fundamentals course.
B1287LL/A Regular price $1095.00 (U.S.)
Special price $695.00 (U.S.)

APDA Top Ten
The following were the APDA top-ten selling products in June1993

1. E.T.O. Starter Kit
2. MacTCP version 1.1.1 Developer’s Kit
3. Macintosh Programming Fundamentals version 1.0.13.

4. MPW C/C++ CD Bundle version 3.2.3
5. Macintosh Common Lisp
6. DAL VAX/MVS Server
7. QuickTime for Windows Developer’s Kit version 1.0
8. QuickTime for Macintosh Developer’s Kit version 1.5
9. MacApp
10. MacNosy

Ordering Information

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

The indicates the trade shows/events at which Apple Computer, Inc. is scheduled to exhibit as of press time. This list may be incomplete. If you have information about a show that you want listed here, contact Developer Technical Communications, 20525 Mariani Avenue, M/S 75-3B, Cupertino, CA 95014.
For further information check the Events folder on AppleLink (path—3rd Party Connection:Events).

June 21 through 23
French Developers
Conference
Deauville, France
Contact: Catherine Massot
AppleLink: FRA.PROMO
33-1-6986-3620

June 23 through 25
 Digital World
Beverly Hills, CA
Contact: Julie Marquette
AppleLink: MARQUETTE1
(408) 974-3664

June 28 through 30
 NECC
National Educational Computing Conference
Orlando, FL
Contact: Sue Collins
AppleLink: COLLINS3
(503) 346-3537

June 29 through July 1
PC EXPO
New York, NY
Contact: Jeryl Gerhardt
AppleLink: JERYL
(408) 974-2368

July 1 through 2
Japan Developers Conference
Tokyo, Japan
Contact: Liz Marron
AppleLink: LIZ.MARRON
(408) 974-4575

July, 28 through 31
NACS
Seattle, WA
Contact: Jeryl Gerhardt
AppleLink: JERYL
(408) 974-2368

August 3 through 6
 Macworld
Boston, MA
Contact: Dave Billmaier
AppleLink: BILLMAIER1
(408) 974-4371

September 7 through 15
 IPEX ‘93
Birmingham, UK
Contact: Francine Rose
AppleLink: ROSE9
44-81-730-2481

September 15 through 18
 Apple Expo Paris

Paris, France
Contact: Catherine Massot
AppleLink: FRA.PROMO
33-1-6986-3620

September 20 through 24

 Data Kontor

Stolkholm, Sweden

Contact: Per Hedlund

AppleLink: HEDLUND1

46-8-703-3079

AppleDirections

Volume 1, Number 2

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

Editor
Paul Dreyfus (AppleLink: DREYFUS.P)

Technical Editor
Gregg Williams (GREGGW)

Business & Marketing Editor
Dee Kiamy (KIAMY)

Designer
Robert Stone

Production Editor
Lisa Ferdinandsen (LISAFERD)

Contributors
Juan Bettaglio, Cassi Carpenter, Suzanne Dills, Sharon Flowers, Chloe Freeland, Andie Galt, Mark Johnson, Stacy Moore, Rilla Reynolds, Caroline Rose, Anne Szabla, Wendy Tajima, Jessa Vartanian, Liz Westover,

Manager, Developer Press
Dennis Matthews

Manager, Developer Support Information
Greg Joswiak

Production Manager
Diane Wilcox

PrePress/Film

Aptos Post

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