Inside this issue

NEW *APPLE DIRECT* Editor Paul Dreyfus introduces himself.

(see News folder)
***

APPLE HAS RELEASED A/UX 3.0, its operating system for the enterprise computing market.

(see News folder)
***

ROMin HOLIDAY, this month's CD, contains the usual plethora of material for developers, including International Market Guides and Apple International contacts, C++Tools, and an interactive supplement to the *Apple Human Interface Guidelines.*

(see News folder)
***

***IT SHIPPED!* LISTS** new and updated third-party products shipped in May 1992.

(see News folder)
***

THE HOT PRODUCTS for May are three debuggers; also read about products recently made available by APDA.

(see News folder)
***

WE PICK UP OUR coverage of human interface issues with an article about how constraints can help reduce the complexity of using System 7.

(see News folder)
***

ONE EFFECTIVE WAY of limiting unauthorized software use is network licensing. Recently, a variety of computer firms have proposed a new network licensing standard.

(see News folder)
***

BUSINESS & MARKETING

NEED TIPS on how to create effective demonstrations of your software? The president of a company that designs sales and training demos tells you some dos and don’ts of demo design.

(see Business & Marketing folder)

***

TERRY FLEMING of Timeworks writes about creating a user group marketing program that pays for itself.

(see Business & Marketing folder)

***

YOU CAN STILL purchase tapes of sessions from the 1992 WWDC.

(see Business & Marketing folder)

***

THE MEXICO Developer Seminar, cosponsored by Apple and MACWORLD magazine, will be held in July in Mexico City.

(see Business & Marketing folder)

***

Apple Announces Newton PDA Technology

The news is out: Apple does more than make computers. At last month's Summer Consumer Electronic Show in Chicago, John Sculley made the first public announcement about Newton, a line of "personal digital assistant" products that will become available starting in early 1993. No specific information on developer opportunities on PDAs (which Apple expects to be limited initially) will be available until later this year, but the May 29 announcement revealed a lot about Apple's future plans.

Apple describes "personal digital assistants" as devices that use digital technology to bridge the gap between personal computers and consumer electronics. Newton is the first such PDA technology; it comes from Apple's new Personal Interactive Electronics (PIE) division, which was organized to extend the company into new growth areas where Apple has unique technology and business advantages.

According to Apple's Chairman and CEO John Sculley, "The convergence of digital technology and information represents the biggest opportunity for Apple and other vendors in the personal computer, consumer electronics, telecommunications, entertainment, and publishing industries since the advent of personal computers."

Sculley went on to say, "Apple is poised for success with multiple projects and technologies coming to fruition over the next several years. Newton is a technology which exemplifies the best of Apple's strengths—using leading edge software technology to allow people to do things more easily and more efficiently."

Newton is an entirely new technology from Apple that will be the basis for a broad array of new products. To assist in the proliferation of Newton technology, Apple is licensing Newton to selected vendors to use in their own version of Newton devices. Last March, Apple announced such a relationship with Sharp Corporation of Japan. Sharp has licensed Newton technology from Apple and will also jointly design and produce the first commercial product version of Newton technology for both companies, due out in early 1993.

The first Newton products will be electronic notepads that intelligently assist the user in capturing, organizing, and communicating ideas and information. These products will be small, portable devices that allow the user to do freeform notetaking, drawing, calculating, scheduling, and communicating. Newton's

revolutionary new RISC (reduced instruction-set computer) hardware technology will offer performance capabilities similar to those of a high-end personal computer. And because of its unique software environment, Newton products will be very easy to learn and use.

New Breakthrough Technologies. Apple has spent over four years developing the groundbreaking technologies that are at the core of the Newton product line. They include:

•*Newton Intelligence:* Newton products will actively assist users in their day-to-day tasks. As a user uses a Newton device, it will learn more about him or her and actually propose more efficient solutions. For instance, if a user wanted to schedule lunch with Jane, he or she would simply write "lunch Jane Thursday." Newton technology would know that "lunch" normally means 12 P.M. (noon), that "Jane" means the Jane Green who is listed in the user's address book, and that "Thursday" probably means this Thursday. A Newton device would then suggest this to the user by opening its calendar to Thursday and scheduling lunch from 12:00 to 1:00 with Jane Green.

•*Recognition Architecture:* The goal for the recognition architecture is to make using Newton products as easy as using pencil or pen and paper. Newton products will be able to read the users' writing, transform it into text as they write, and quickly configure and scale drawings or sketches. Newton technology does not require users to print letters in boxes or write only on lines on the screen's surface; they can write in a natural, freeform manner. Newton technology is different from other pen-based operating systems because it is based on the simultaneous use of several recognition technologies. The benefit to users is a higher recognition factor and greater adaptability to personal style.

•*Information Architecture:* A Newton device will be a single repository for all of the little pieces of information that users would typically accumulate in various forms (for example, phone numbers, business cards, directions, meeting notes, and birthdays). Because of the advanced object-oriented data structures that Newton products will use, users will be able to organize data so that they can easily access it in a variety of ways.

For instance, a user could view all the information that relates to a specific customer, a specific week, or a specific person.

•*Communications Architecture:* Newton technology was designed from the ground up to take great advantage of communications, which Apple sees as

being a key technology for the 1990s. These new products will make it easy for Newton users to communicate in meetings, on the street corner, or during lunch. Newton devices will have built-in wired and wireless communications capabilities.

For instance, two Newton users could compare calendars or exchange electronic business cards. On their own, Newton users could fax a letter, check electronic mail messages, or connect to a satellite news service to obtain current news or stock information.

•*Hardware Architecture:* Newton technology is based on a new breed of powerful RISC processors optimized for high performance, low power consumption, and low cost. The Advanced RISC Machines (ARM), Ltd. RISC processor that Apple has chosen, the ARM 610, gives Newton products the equivalent power of desktop computers while consuming less battery power than the average flashlight. The ARM 610 features a special memory management unit designed by Apple and ARM Ltd. to be optimized for affordable, low-power-consumption, high-performance systems like Newton.

Industry Leaders Announce Support. The first group of major companies has announced support for Newton technology and has committed to developing future complementary products. Representing many industry segments, the range of products expected will be focused in three general areas:

•*Communications products*—to enhance the use of Newton devices in a mobile environment

•*Content products*—to offer users a wide spectrum of interesting and personal information for their Newton products

•*Compatibility*—to allow users to use Newton devices easily in parallel with existing computer systems

Motorola, SkyTel, Random House, Traveling Software, Bellcore, and Pacific Telesis all made announcements on May 29 in conjunction with Apple.

Availability. The first Newton product from Apple will be an English-language version available in early 1993. Newton-based products from Sharp Corporation are also expected to be available in the same time frame. Apple will announce prices at the time of delivery.

****

What Newton Means to Developers

With a new kind of product comes a new way of doing things. As described in the accompanying article, the Newton product family is significantly different from the Macintosh. John Sculley mentioned during his keynote talk at the Worldwide Developers Conference in May that Apple's PDA business will be a separate business from the Macintosh business. This means that the PDA business will have a different business model from the Macintosh business, including how Apple interacts with third-party developers.

Personal interactive electronic products like Newton are often limited in scope (compared to personal computers), which means that the opportunities for third-party developers may also be limited.

As Apple develops this product line and set of technologies, *Apple Direct* (and other channels) will make sure you know about emerging developer opportunities. We expect opportunities to be limited initially, expanding over time as the product line matures and broadens.

Macintosh Quadra 950 Introduced

Apple introduced its most powerful Macintosh computer last month, the Macintosh Quadra 950.

Built around a 33-megahertz (MHz) Motorola 68040 processor, the floor-standing Macintosh Quadra 950 provides 30 percent higher performance than the Macintosh Quadra 700 and 900 models. The Macintosh Quadra 950 ships with 8 megabytes (MB) of memory, which can be upgraded to 64 MB. A new, high-performance 230 MB internal hard disk drive replaces the 160 MB disk drive provided with previous models.

The Macintosh Quadra 950 outperforms its competition. In benchmark tests using productivity applications developed for both Microsoft Windows and the Macintosh, Ingram Laboratory found that the Macintosh Quadra 950 was faster than many 33-MHz and 50-MHz 486-based computers.

The Macintosh Quadra 950, along with the rest of the Macintosh Quadra line, features built-in, high-performance connections to Ethernet and LocalTalk. For Ethernet connections, the Macintosh Quadra line is equipped with an Apple Ethernet port and a media-independent connecting system that allows customers to use any standard Ethernet media, including thin coaxial, thick coaxial, and twisted-pair wiring.

The Macintosh Quadra 950 also provides workgroups with their best Macintosh server yet. In addition to large internal storage capacity, a fast processor, and a fast dual-SCSI architecture, the Macintosh Quadra 950 has an I/O clock that runs at 25 MHz (up from 16 MHz for Macintosh Quadra 700/900), enabling increased Ethernet and LocalTalk throughput.

Its standard configuration (with 8 MB of memory and a 230 MB internal hard disk drive) enables it to run without modification A/UX 3.0, Apple's complete integration of System 7 and the UNIX® operating system.

The Macintosh Quadra 950 supports 16 bits per pixel on all color displays up to 21 inches, which permits near true-color resolution and superior image clarity without the memory and time overhead of 24- or 32-bit pixels.

Macintosh Quadra 900 owners can upgrade to the Macintosh Quadra 950 by having a new logic board installed by an Apple authorized reseller. In the United States, Apple will offer special pricing to users upgrading from the Macintosh Quadra 900 to the Macintosh Quadra 950. Apple expects the upgrade kits to be available in the U.S. in early July.

In addition to introducing the Macintosh Quadra 950, Apple also reduced the U.S. manufacturer’s suggested retail price of its Macintosh Quadra 700 computers by up to 8.8 percent, effective immediately in the U.S.

For Developers Interested in Native PowerPC Applications

At the 1992 Worldwide Developers Conference, Apple began talking with developers about PowerPC, a new RISC (reduced instruction set computer) processor that we will use to create a new family of Macintosh computers. Under emulation, existing Macintosh software will run on it without changes, but developers will have several ways to deliver software that will run at native speed on PowerPC Macintosh computers: binary to binary translation, recompilation, and new application development.

If you are interested in these possibilities, send a letter to:

Apple Computer, Inc. (PowerPC)
20400 Stevens Creek Blvd.
MS 75-7E
Cupertino CA 95014

Please include descriptions of your current and future products, including company and contact information, primary development environment(s) and language(s), and use of programming frameworks, third-party libraries, and floating-point data types.

Antipiracy Technologies: Some Modest Proposals

Technology Can Help, But It's No Panacea

by Gregg Williams,
Apple Direct staff

Software piracy is costing you money—you know that. You've priced your software to be as affordable as possible, but people still pirate it. Worse yet, the easier you make it to use, the more likely they are to pirate it. You know that your legitimate customers don't like software or hardware antipiracy add-ons, but you have no choice. Such things *are* the one sure way of protecting your software from nonpaying users. Aren't they?

Well, it depends. Welcome to the gray no-man's land of using technology to combat software piracy.

Technology alone is not the solution to software piracy. In the long term, user education and nontechnology responses to piracy (such as support to registered users and good printed documentation) will make the biggest difference.

Unfortunately, software piracy is hurting your company today, and creditors don't accept "the long term" as payment for their bills. This is where antipiracy technologies step in: They aren't the total solution but—depending on the situation—they may be part of it. The most important step is to decide what to use, and when.

Five Antipiracy Messages

The thrust of this article is to describe *antipiracy technologies*—that is, things that you can do to your product (or ways of limiting access to it) that will decrease the number of unauthorized users (that is, people who use it without paying for it). Here are five important points you should keep in mind when considering which technology, if any, you should use:

• *There are no magic bullets*. It's sad but true. No one technology is going to solve the entire problem of software piracy. Almost every technology-based antipiracy strategy can be defeated or circumvented by someone who is determined to do so. Many strategies can stop or deter piracy in one situation

but not another. Because of this, every antipiracy strategy is only a partial solution, and you will probably want to implement several strategies in parallel.

• *A good strategy will contain both technology and nontechnology components.* We covered the major nontechnology antipiracy strategies in "The Ways and Means to Fight Piracy," by Karen Wickre, in the March 1992 *Apple Direct.* You should read that article so that you'll know *all* possible options before you decide which strategies (note the plural) to use. It's easy to throw technology at this problem, but it's probably a mistake to limit yourself to technology alone. Which brings us to the next point....

• *In general, the more rigorous your antipiracy efforts are, the more objectionable they will be to your legitimate users.* Every antipiracy strategy has negative aspects, and the last thing you want to do is alienate your *legitimate* users. One important rule of thumb is that nontechnology antipiracy efforts are usually less objectionable than efforts that use software, which are in turn less objectionable than efforts that use hardware.

• *People are using software in new ways, and your antipiracy strategy needs to acknowledge this and respond to your customers' needs.* With networks, PowerBooks, and low-cost Macintosh computers becoming commonplace, people are using their Macintosh computers in new ways, and they gravitate to software (and software policies) that help them to do so. *Concurrent licensing*—that is, specifying the maximum number of copies of a program that can legally be running at the same time—is a key issue related to network licensing, which Apple sees as a promising antipiracy technology. (See "Network Licensing," later in this article, for more details.)

• *The easier you make it for people to use your software legally, the more likely they are to do so.* Many users, especially corporate users, want to "do the right thing" but can't be bothered with any antipiracy tool that makes their lives more complicated. It's better to give them a simple tool that prevents some software piracy than a complicated, more effective tool that they don't use at all.

Factors to Consider. You need to keep your product, your market, and your customer firmly in mind before you look at specific antipiracy technologies. For example, if your product costs $95 and needs minimal documentation, you should use a different antipiracy strategy than you would for a $1500 real-estate management program that requires a great deal of printed documentation.

Consider your intended audience. Do your customers work alone or with others? How likely are they to need customer support? How will they react to the antipiracy strategy you plan to implement?

The environment in which your users work is equally important. Are they isolated users or are they connected to a network? Will they be working at home, at the office, or both? Do they share a Macintosh computer with others? Do they work on more than one Macintosh in the course of a day?

With more and more PowerBooks in the Macintosh installed base, users are more mobile than ever. Do they use their PowerBooks on the road? In other people's offices? Might they want to borrow a PowerBook and use it?

You will also want to take a hard look at how much an antipiracy strategy costs. Is the per-unit cost of the technology affordable relative to the software's retail price? Are there any hidden manufacturing costs? Does the technology incur ongoing administrative costs? Currency costs aside, what will this technology cost you in customer loyalty?

THE GOOD, THE BAD, AND THE POSSIBLY USEFUL

Apple has a continuing commitment to help make your company financially successful, and one important way of doing this is to help you reduce the money you lose to software piracy. The Apple Developer Group created a task force to study software piracy and recommend potential solutions.

This article contains the ADG task force's recommendations for technology-based antipiracy strategies. Some make a lot of sense, others are useful in some situations, and still others are usually not a good idea, but we still list them here in case your situation warrants their use. For each type of technology, Table 1, shown below, lists companies that make Macintosh-related product(s) . We'll devote the rest of this section to those solutions that Apple thinks have the best chance of being effective.

Personalized Software. It's relatively simple to modify your program so that when users run the program for the first time (or, alternatively, when they install it onto a hard disk), they enter some personal information about themselves that then reappears at some time during the normal use of the program. (Most such schemes ask for one's name, and sometimes the company name, and then display that information every time that person launches the program. Seeing

this makes most users think twice about casually distributing the program to others.)

This antipiracy strategy, which we call the *personalized software* strategy, is not foolproof; any user who wants to get around it *can* get around it. However, this strategy has two things going for it. First, it presents a slight obstacle that keeps the average person honest. Second, it's fairly simple to add to your program and, once implemented, it costs you nothing. (In other words, you don't have to pay another company to use its antipiracy software or hardware.)

Document Translation. Many people steal a program just to print out documents in that program's unique format—and once the program is on their disk, they may start using it. So you, the developer, can probably prevent *some* piracy by making it easy for other programs to open your programs files. That way, users can print the document out, cut and paste data from it, and perhaps even change and translate it back to your program's format—all without pirating your software.

Currently, one of the most ubiquitous file translation systems is the XTND system, designed by Claris Corporation. If you're interested, you can buy the XTND Developers Kit version 1.3, which is available through APDA. (To order from APDA, see "Now Available From Apple" in the News folder.) If you create and distribute an XTND filter for your program's document, others can use any XTND-aware program to open documents in your program's format. And that may reduce the number of people pirating your program.

Further down the road, Apple is working on something that will help users open documents they don't have the program for—it's called the Translation Manager. Normally, when the user asks a program to open a file, the program presents a dialog box containing all the files it knows how to open. When the Translation Manager is present, it will add to the dialog box all the files that it can translate into a format that the program can open. (It does this totally without the program's knowledge, so the Translation Manager assists *every* program, not just ones that "know" that it's present.) Look for more details on the Translation Manager in a future issue of *Apple Direct.*

Limited-Use Programs. Let's face it—when it comes to convincing customers that your program is the right one for them, nothing beats hands-on experience. Every program has its own personality, and most people

(understandably) won't part with $400 or more without a look at the program. Unfortunately, the only way most people can get a preview is by "borrowing" a copy from a friend, and you know how hard it is to delete a file once you've gotten it onto your hard disk.

It may make sense for you to create a *limited-use program.* This is a version of your program that lets users fully experience your program—or at least enough of it that they'll want to go out and buy the real thing.

You can create a limited-use program in several ways. You might make the program so that it works, say, until a set date or the week after it's first launched. You might make it work for small documents but not for large ones—a word processor that can only create two-page documents, for example, or a database that can have only a hundred records. Or perhaps you make the program print each page with the words *Demo Copy* across the page. These are just examples—you can figure out what makes the most sense for your program.

However, you should pick a limitation the user can live with. We've found that many businesspeople don't like "crippled" programs. And *nobody* likes programs that won't print documents. (Programs that won't save files come in a close second.) If the limitation annoys users or is too intrusive, they'll focus on that and won't get the rush that comes from using a program that they absolutely *must* possess. Be sure you know your customer before you implement this strategy.

Equally important, you should pick a limitation that *you* can live with. One of the drawbacks of limited-use programs is that some users may find them adequate, and they won't need to buy the full, working version of your program. If you choose the limitation carefully and provide other things that the user wants (like good documentation, support materials, and customer service), the number of users who buy the fully functional version of your program will far outnumber the potential sales that you might be losing.

One thing you should be sure not to do is to create a "time bomb"—that is, a program that stops working without the user's advance knowledge. Using time-bomb programs makes sense only when you're distributing alpha or beta software for testing; if the software stops working, testers know that they should contact you to get the most recent version of your program. (Even then, it's *still* a good idea to warn testers of the program's expiration date.)

Save-a-Copy. Once you have created a limited-use program, you can send it to user groups, put it on electronic bulletin boards, and hand it out freely. After all, you *want* people to copy this software. With one simple extension to the work you've already done, you can make an important addition to your network of distribution: your customers.

The idea of save-a-copy is simple. In a menu of your installation program or the program itself, add an item that says "Make a Copy." When the user selects this menu item, your program then creates a limited-use copy of itself on a floppy disk. That way, when a potential customer asks a coworker, "Say, that's a nice program—can I have a copy?" the person being asked can give the copy away without worrying about ethical issues. If the coworker really likes your program, he or she will be the kind of "insider" salesperson that money can't buy, and the limited-use copy given away may bring you one new sale, maybe more. (For an avant-garde extension to this idea, see "O Brave New World" below.)

BAD PROGRAM! BAD!

Before we cover those antipiracy solutions that may be appropriate in your situation, I'll mention briefly one technology that Apple feels you should use only as a last resort: *floppy-disk copy protection.* (Please keep in mind that I'm using "bad" in a relative, not an absolute, sense. You should use whatever antipiracy strategy makes sense for you, but you should also be aware of the tradeoffs involved.)

Floppy-Disk Copy Protection. This strategy involves encoding certain floppy-disk sectors in a way that makes it difficult for someone to copy the disk. (You usually do this to the floppy disk on which you distribute your program, also called a *distribution disk*.) In many cases, an installer program that you supply allows users to install a limited number of copies of the program onto a hard disk. In some implementations, users can also make additional copies of the program, but when launched, these copies will not work unless the original floppy disk (also called a *key disk*) is present.

Floppy-disk copy protection *does* limit software theft, but Apple recommends that you consider using it only as a last resort. Since this kind of copy protection depends on specific characteristics of the hardware, the copy protection may occasionally malfunction, thus preventing your customer from legally using the

program he or she bought. Also, you have no way of knowing whether the floppy-disk copy protection that you use will work with future hardware from Apple and other companies.

Technical reasons aside, floppy-disk copy protection is bad because it breaks the most important law governing antipiracy technologies: *Thou Shalt Not Annoy the Legitimate User.* Limited-use installer programs and key disks are inconvenient to use, and they often cause legitimate users problems. When this copy protection malfunctions, you suddenly have a *very* unhappy customer, one who sees copy protection as a useless encumbrance that doesn't deter the software pirate but that penalizes the *legitimate* user for being honest.

(In my opinion, one version of key-disk copy protection is, in certain situations, defensible. Suppose you supply your program on CD-ROM, and it refers to data that legitimately needs to be on the CD-ROM because it's too big to put on a hard disk. This gives you automatic protection against software piracy because most people aren't able to copy a CD-ROM. You may be tempted to have your program periodically read from the CD-ROM just to ensure that only the possessor of the CD-ROM can use your program. Think carefully before you use this tactic—users are more likely to resent any *artificial* tie between the program and the CD-ROM.)

POSSIBLY USEFUL STRATEGIES

The technologies described in this section are "moderate" in several ways. They are more effective than the "good" technologies and less effective than the "bad" ones. They are also more intrusive than the "good" technologies but less intrusive than the "bad" ones. You should carefully consider your needs to see if the technologies described here might be useful to your program and your market.

Software Password/Serial Number. In this antipiracy technology, the user must enter a password or serial number into your program the first time that he or she runs it. You must customize each distribution disk to require a unique password or serial number. This technique, when used with personalization and registration, provides a strong deterrent to software theft without seriously inconveniencing legitimate users.

Software serial numbers (I'll use the term to include passwords) do not prevent someone from copying your program and giving it to someone else. But if someone copies the original distribution disk and hands it casually to a third party, that person, possessing the program but not the serial number, will not be able to launch it. Furthermore, if your program displays the same serial number that appears on your customer's registration card, most people are unwilling to let others have copies of a program that can be traced back to them.

Single-User Activation String. Though you might not know this method by name, you're probably familiar with it—several major companies use it to distribute hundreds of fonts on one CD-ROM, later selling users the *activation string* that "unlocks" selected fonts by means of a telephone transaction paid for by a credit card. The mechanics of the transaction ensure that the activation string works for that transaction only, so that no one can use the string to unlock any other fonts.

The activation-string strategy works best with files that need no documentation (like fonts and clip art) or with relatively simple programs that contain built-in documentation. The activation-string technology works on both CD-ROMs and floppy disks. When used on CD-ROMs, it gives you an economical method of distributing huge amounts of software (over 600 megabytes) while still ensuring that people use only those products that they pay for.

The activation-string technology is not perfect, however. Either you or some third party (often the vendor of the activation-string technology you use) will have to provide the needed telephone support, maintain the database of registered users, and collect and distribute the incoming payments. (In some cases, the vendor of the activation-string technology will, for a fee, administer this process for you.) On the user's side, this technology is less than perfect because the unlocking process is cumbersome and sometimes confusing.

Activation-string software has two more disadvantages. The first is quite real: once your customers have bought your software and unlocked it, they are then able to share it illegally with others. The second disadvantage is more theoretical: if someone breaks the security mechanism within the activation-string technology, they suddenly have access to *all* the software on the distribution medium (which may be dozens or *hundreds* of products if the software is on a CD-ROM).

One important aspect of activation-string technology is that it requires you to rethink the way you package, document, and distribute your software. Realize that you're distributing your software directly to the user—in most cases, there are no store salespeople to make customers aware of your software. Since your product is totally electronic, it is "faceless"—that is, it has no packaging or visual style that customers can easily remember.

Networked Serial Numbers. If you know your program is likely to be needed by multiple users on a network, you can use this method to prevent all the users from sharing the same copy. When any copy of your program launches, it "listens" for its own serial number, displays a message to the user (something like "This copy of GerbilCalc is already being used by George"), and exits. This prevents casual sharing of your program, but it annoys users. If you implement this strategy, be sure that you don't degrade network performance by tying it up with serial-number packets.

Documentation-Key Software. This strategy is almost in the "bad" category of antipiracy strategies. Though it's most often seen on computer games, you occasionally see it elsewhere.

It usually works like this: Every time the user launches the program, it requires him or her to locate a random datum in the documentation and enter it into the program. (The program often tells the user something like "Enter word 3 on line 6 on page 28 of the documentation." If you want to increase the effectiveness of this method—at the risk of alienating your customer—you can print the needed information on dark red paper that can't be successfully photocopied.)

Though the documentation-key method eliminates most casual copying (its effectiveness is proportional to the difficulty of photocopying the needed information), the method has several drawbacks. First, it really annoys legitimate users. Second, it prevents customers from using the program at both home and work; they can't use the program at both locations without remembering to carry the documentation home with them. This limitation is particularly burdensome to customers who love the portability of their PowerBooks. (Imagine having to carry, say, a dozen instruction books with you all the time so you can use the PowerBook and your document-key-protected programs.)

Dongles. These hardware devices (don't ask me how they got their name) are small boxes that connect to the Macintosh via some external port; the program being protected occasionally checks the port and halts if the dongle isn't present. The underlying premise is simple: software is easy to duplicate, but custom hardware is not. If your program checks for the presence of the dongle (either once or periodically, and in a way that an unauthorized programmer can't circumvent), the dongle becomes a "key" that limits the number of users of your software—it's "one dongle, one user," and there's no way to get around it.

Dongles also have the same technical and usability problems as floppy-disk copy protection schemes. Since dongles are hardware, they also present physical problems—namely, if you have, say, six programs that require separate dongles, will you have the physical room to daisy-chain them off the same port? (And what if they don't work with each other?)

Two more disadvantages come to mind. Users should turn their Macintosh computers off before adding or removing dongles; this is a inconvenience that no one likes. Also, dongles usually come from third parties that make their business from supplying this kind of technology. By using dongles, you will add a significant amount to the unit cost of your software. Dongles are most often appropriate for high-price, low-volume applications that run by themselves on a dedicated Macintosh.

You can design dongles to work off any Macintosh port. If you must use them, however, Apple recommends that you use the Apple Desktop Bus (ADB) port, not the serial or SCSI ports.

END-USER SOLUTIONS

Some of the most promising antipiracy technologies around aren't really meant for you, the developer. Instead, they are meant for your customers. I mention them here because you may want to encourage your customers to use them. A page in your documentation devoted to a nonjudgmental discussion of you and your customers' responsibilities to each other will educate your customers and help reduce software piracy.

Audit Tools. As I said earlier, the easier you make it for customers to comply with the law, the more likely they will be to do so. Once they understand that they—and, more importantly, the staff they manage—must own legal copies of all the software they (or their staff) use, they must then find out exactly *what* is on

their (or their employees') hard disks. Several programs exist that make this task easier.

Apple has worked with the Software Publishers Association to create a Macintosh version of SPAudit, a program that searches through the user's hard disks and makes a list of all the programs it finds. This is a good tool for a single user or a small group, and the price is right—it's free, and you can give it to anybody. You can get the SPA Self Audit Kit (which includes the SPAudit program) by contacting the Software Publishers Association; see table 1 for details. You can also get SPAudit from *ROMin Holiday,* the June 1992 Developer CD, and it's on AppleLink under the Developer Support icon.

If users need more control over the software that their networked staff uses, several commercially available products can do the job. Not only can they report on the contents of users' hard disks, they can also update every copy of a program (which is always a headache for administrators) and perform other useful functions. See Table 1 for companies that supply such products.

Network Licensing. This method *works*—it's in use at Apple and several other major corporations, and your customers, regardless of their corporate size, can use it to control legal access to almost any program used on a network. When this system is in place, users can freely copy a program, but when they launch it, special code in the program checks with a "key" server that doesn't let the program run if other users are already running the maximum number of copies that the network's owner is legally entitled to use. A user who has been denied use of the program can request to be notified when someone else exits the program.

Network licensing works, but it depends on a concept you may not have given much thought to: *concurrent licensing*. Traditionally, companies have licensed their software to be used on a single computer or, more recently, by a single user. Concurrent licensing grants the use of a program to a set number of *simultaneous users*. This approach makes a lot of sense in a networked office environment. On the customer's side, it's a lot more practical than having to buy one copy for each computer or user. (If you were a manager of 50 employees, would you buy 50 copies of a $400 spreadsheet if your employees used it primarily to do a monthly status report?)

More important to you as a developer, concurrent licensing makes sense because it is *enforceable.* Even if you sell those 50 copies of your spreadsheet,

you can't guarantee that the 51st or 60th or 70th employee will buy a new copy. But if your customer uses a network-licensing program and pays for the right to run (for example) 20 simultaneous copies, you know with reasonable certainty (based on your trust that the customer *wants* to use software legally) that you are getting paid for a certain level of usage. If your customer's department grows to 100 and employees are routinely denied access to your spreadsheet, your customer will probably come back to you and pay for the right to run, say, 40 copies simultaneously.

However, your customers can't use third-party network licensing products unless you establish concurrent licensing as a way of legally using your programs. Network licensing makes both installation and compliance easy. (Apple recently joined the Software Publishers Association and about 20 companies in defining an application programming interface (API) that would help standardize access to network-licensing technologies. See "License Service API Promotes Acceptance of Network Licensing" in the News folder.)

Is network licensing *the* solution to your antipiracy problems? No—current network-licensing products work only in networked environments. They do not address the issue of single-user software piracy, nor do they help networked users when they want to use your programs at home or on a PowerBook. (And network-licensing products *can* be circumvented, although usually not by the average user.) But if many of your customers use your programs with a network present, you should seriously consider adopting a concurrent-use policy and educating your customers on the advantages of network-licensing products.

WHAT'S YOUR STRATEGY?

Someone once said that for any situation, no matter how complicated, there's always a simple answer—and it's wrong. I can't think of a situation for which this is more true than that of your personal antipiracy strategy. There is no perfect answer—every strategy is a tradeoff between security and user unfriendliness. Any good overall solution will include both technology and nontechnology strategies.

You are not alone in your situation, and perhaps talking to others will help you make better decisions and assure you that you're not forgetting anything important. You should use the Developer Talk section of AppleLink (path—Developer Support:Developer Talk:Anti Piracy Discussion) to talk with other developers and express your opinions. Since Apple people are reading and

contributing to this area, this is also the place for you to share your concerns on software piracy and tell Apple how it can help you.

The SPA estimates that Macintosh software developers lost $535 million from piracy in 1990. Unless you're in a very special situation, you are losing money through lost sales of your software. You cannot afford *not* to analyze your situation, even if only to decide that you don't need to do anything (or that any antipiracy strategy would cost more than it was worth). I hope that this and previous articles in *Apple Direct* will speed this process for you.

This table lists known antipiracy products and their vendors as of mid-May, 1992. This list is for your information only. Apple does not endorse any of the companies or products listed here.

This table includes fax telephone numbers and AppleLink addresses, where available.

DONGLE HARDWARE

Eve

Rainbow Technologies
9292 Jeronimo Rd.
Irvine CA 92718
Phone: (714) 454-2100
Fax: (714) 454-8557

Macintosh Memory Key

ProTech Marketing, Inc.
9600-J Southern Pine Blvd.
Charlotte NC 28273
Phone: (704) 523-9500
Fax: (704) 523-7651

Mactivator

Software Security
1011 High Ridge Rd.

Stamford CT 06905
Phone: (800) 333-0407
Fax: (203) 329-7428

SecuriKey
Micro Security Systems, Inc.
150 Wright Brothers Drive
Suite 560
Salt Lake City UT 84116-2847
Phone: (800) 456-2587
Fax: (801) 575-6621

SINGLE-USER ACTIVATION KEY

Access
allmedia
940 6th Avenue SW, Suite 1150
Calgary, Alberta
CANADA T2P 3T1
Phone: (403) 261-7660
Fax: (403) 237-6626

Windmill
MacVONK•USA
313 Iona Ave.
Narberth PA 19072
Phone: (215) 660-0606
Fax: (215) 668-4360
AppleLink:

AUDIT TOOLS

DiskAuditor
Expert Systems, Inc.

2616 Quebec Ave.
Melbourne FL 32935
Phone: (407) 242-0140
Fax: (407) 952-5605

GraceLAN 2.0
Technology Works
4030 Braker Lane West
Suite 350
Austin TX 78759
Phone: (512) 794-8533
Fax: (512) 794-8520
AppleLink: TECHWORKS

netOctopus
MacVONK•USA
(See above)

NetWork SuperVisor 2.0.1
CSG Technologies
530 William Penn Place
Suite 329
Box 131
Pittsburgh PA 15219-1820
Phone: (412) 471-7170
(800) 366-4622
Fax: (412) 471-7173

Radar 2.0
Sonic Systems
333 West El Camino Real
Suite 280
Sunnyvale CA 94087
Phone: (408) 725-1400
Fax: (408) 736-7228
AppleLink: SONIC.SYS

Status•Mac 2.0.2

ON Technology, Inc.
155 Second St.
Cambridge MA 02141
Phone: (617) 876-0900
(800) 548-8871
Fax: (617) 876-0391
AppleLink: ON.TECH

Network SuperVisor 2.0
CSG Technologies
(see above)

SoftLibrarian
Jeddak Software
2540 North First St.
Suite 301
San Jose, CA 95131
(800) 982-6900
Fax: (408) 894-9020
AppleLink: JEDDAK

MacSupervisor™
Hi Resolution Ltd.
4 Smallbridge Cottages
Horsmonden
Kent
ENGLAND
TN12 8EP
Phone and fax: 0580 211194
AppleLink: WIZZARDWARE

NETWORK LICENSING

KeyServer 3.0
Sassafras Software
PO Box 150
Hanover NH 03755
Phone: (603) 643-3351
AppleLink: SASSAFRAS

Quota 2.0
Proteus Technology
9919 68th St.
Edmonton, Alberta
CANADA T5A 2S6
Phone: (403) 448-1970
Fax: (403) 428-6882

PUBLICATIONS AND PRODUCTS FROM ASSOCIATIONS

Guide to Software Management
Business Software Alliance
1201 Pennsylvania Avenue NW
Suite 250
Washington DC 20004
Phone: (202) 737-7060
Fax: (202) 737-7063

Network Software Licensing White Paper
Microcomputer Managers Association
PO Box 4615
Warren NJ 07059
Phone: (908) 321-4701
Fax: (908) 321-3905

Network License Survey
Self Audit Kit (includes SPAudit)

Software Publishers Association
1730 M Street NW
Suite 700
Washington DC 20036
Phone: (202) 452-1600
Fax: (202) 223-8756
AntiPiracy Hotline:
(800) 388-7478

****

O Brave New World

You don't have to be a marketing consultant to recognize that we're in an entirely new world when it comes to how people buy and use software. Here are a few outrageous ideas that may no longer be, well, outrageous.

Make Software Piracy Work *For* You. This idea is an extension of the make-a-copy strategy (where you make it easy for users to give a limited-use copy of your program to friends). If you've already added a save-a-copy button or menu item to your program, why not also allow the limited-use copy to print out an order form for itself that includes the name and serial number of the (registered) user who gave the copy away?

The strategy works like this: Let your customers know that they will get a reward if you receive order forms that name them as the original source of the limited-use copy. What should the reward be? That's up to you, but I know that *money talks*—I'd rather receive, say, a check for even as little as $20 than a floppy disk of clip art. A less expensive way to offer your users a cash-like incentive is to reward them with a free or half-price upgrade to the next version of the program.

Here's the beauty of this strategy: Not only do you give your users explicit permission to pass out copies of your program, you also give them a very strong incentive *not* to give out a fully functional copy. (I recommend you make the cash reward as large as you think you can afford. Even if the new sale makes you less money, you will have one more registered user in your files. That automatically increases your installed base and market presence, and the extra names in your database can lead to future software and upgrade purchases.)

Don’t Sell—Lease. Since documentation, customer support, and periodic upgrades are a part of many high-end software packages (like page-layout and computer-aided design programs), why not give your customers the option of paying a smaller quarterly or yearly fee for access to the total package of program, documentation, support, and upgrading?

For example, instead of having users pay, say, $800 for an illustration program, charge them $100 per quarter or $250 per year to use the program. (You can enforce this by having the program expire gracefully after a certain date and adding some way of reactivating the program once customers have paid for additional usage.)

If such a strategy is right for you, your program, and your audience, it has advantages for both your customers and you. They are more likely to try your program out legally instead of pirating it. (After all, who wants to invest a large amount of money in a program without knowing if it’s right for them?) This strategy also gives customers a streamlined, all-in-one solution for purchasing, upgrading, and support. On your side, this strategy gives you a steady, predictable source of income and helps reduce software theft.

New AppleLink Program: Free Bulletin Boards

Apple Online Services (AOS) is launching a new electronic publishing program that will expand opportunities for developers and other companies to reach the more than 44,000 AppleLink subscribers around the world.

Under the program, companies that qualify for Apple's Online Services Information Publishers Program will no longer be charged startup and maintenance fees for bulletin boards, discussion boards, and libraries. Previously, these fees could cost thousands of dollars a year. AOS will continue to charge for AppleLink Extras, which are primarily used for promotions.

Apple Online Services manages Apple's on-line services product line and business, including AppleLink. AOS is currently rolling out a number of initiatives to make AppleLink more broadly available to Apple customers both in the U.S. and in other countries, so that they can take advantage of its on-line information and communications resources.

Publishers participating in the program will still be able to post one- and two-way bul-letin boards, discussion boards, libraries, AppleLink Extras, customized mailing lists to distribute information to select groups, and other options. Developers can also publish information on third-party communications bulletin boards, including the 3rd Party Connection, 3rd Party/Demos & Updates, and News Beat bulletin boards. The range of information that can be published includes product and program announcements, company news, marketing information, pricing and ordering information, and service and technical support information, as well as software updates, upgrades, and product demos.

Developers and other companies can become certified AOS Information Publishers by assigning an individual to manage their information service, follow AOS Information Publishing guidelines, maintain publishing standards, and sign an AOS Information Publisher's agreement. Currently authorized AppleLink Publishers must also sign the new agreement and meet all criteria to qualify.

To enroll, call the Information Publisher's Hotline at (408)974-1300 or contact the AOS manager at your local Apple office. You can also use the application form posted on the Publishing On AppleLink bulletin board located in the AppleLink Help Desk. It also contains more information about the AOS program. Applications will be considered in the order in which they're received, with priority given to developers and third parties whose information services support Apple's business priorities.

A Note From Your (New) Editor

Readers:

It used to be that the editor of *Apple Direct* got personal once a month and wrote an editor's note. Well, I think that was a nice tradition, one that I'd like to pick up periodically again, not only as a way of letting you know what's going on in *Apple Direct* but also to respond directly to readers' concerns and feedback.

This month, what's going on at *Apple Direct* is that I've taken over from Dee Kiamy as editor. First off, I'd like to acknowledge Dee's terrific contribution; when the previous *Apple Direct* editor, Lisa Raleigh, left in January, Dee stepped up from her role as editor of the "Business & Marketing" section to become acting editor-in-chief. Little did she know that her tenure would last four issues, during which time she edited the news in addition to providing her typically great information on business and marketing. She'll continue on as business and marketing editor; thanks, Dee!

You might want to know a little about me: I've been in publishing and editing for about 15 years now. I came to *Apple Direct* from Apple's Enterprise Systems Division, where I was the editor of A/UX documentation. Before that, I worked on networking and communications publications for Apple, first as an editor, then as a manager. I joined Apple after eight years with a book publisher in Boston, where I was managing editor of a 27-volume illustrated history of the Vietnam War. Before getting into books, I wrote for and edited a variety of business newsletters.

I'm excited about putting all my experience together to bring you *Apple Direct* every month. I think it's a terrific publication as it is; needless to say, I'll be bringing my own ideas to the newspaper, and you'll see some changes in coming months as I and the *Apple Direct* staff respond to reader feedback and develop ways to make the newspaper even more useful to you.

Four years of work in Enterprise Systems and Networking and Communications made me pretty enthusiastic about Apple's open systems and business computing efforts; although I'll keep my "bias" in check, I'll do my best to keep you up-to-date on Apple's exciting products in those areas.

It's my priority to be sure that *Apple Direct* gives you the information you need to keep abreast of what's happening at Apple and help you run your business;

tell me how we can do that better, and let me know what you like and don’t like about the newspaper (but, please, be kind).

Send me an AppleLink anytime at APPLE.DIRECT, and happy reading!

Paul Dreyfus

A/UX 3.0: Leading the Way Into Enterprise Computing

Apple has released its operating system for the enterprise computing market—A/UX 3.0, which integrates full System 7 functionality (including QuickTime) and ease of use with the UNIX® operating system.

Underneath the Macintosh user interface, A/UX 3.0 offers an open-systems operating system with true multitasking, standards compliance, the flexibility to run applications from different environments (for example, Macintosh and DOS), connectivity, and advanced computing power. It also provides them with an upgrade path to RISC-based PowerOpen, the open-systems OS under development by Apple and IBM.

A/UX 3.0 gives developers special opportunities, both to reach new markets with existing products and, with A/UX Developer's Tools version 1.1 (due from APDA soon), to develop new products geared specifically for the A/UX architecture.

A/UX 3.0 paves the way for the Macintosh to be used in what Apple calls "enterprise" computing systems—for example, those used by large corporations, higher-education organizations, government agencies—that typically link a variety of computing environments, including Macintosh, DOS, UNIX, and other systems.

With A/UX, users can have simultaneous access to off-the-shelf Macintosh applications in addition to programs the run using UNIX, X Window System, OSF/Motif (from Integrated Computer Solutions), and MS-DOS (using SoftPC from Insignia Solutions). Developers can use A/UX 3.0 to build standards-based software applications and deploy them for end-users on a Macintosh.

A/UX 3.0 provides built-in networking and communications capability that includes both Macintosh and UNIX-based connectivity solutions. The product also includes MacX 1.1.7, the Macintosh X display server, and X11 for A/UX with libraries and a toolkit for X application development.

Applications developed for Macintosh System 7 and A/UX 3.0 will be compatible with PowerOpen, protecting for years to come the investment users and developers have made in software. When available, PowerOpen will be a highly scalable, open-systems architecture available on several computer systems. It will offer developers an easy-to-use, standards-based high-performance operating system that integrates Apple's A/UX and IBM's AIX; it will provide customers with access to the wide range of Macintosh productivity

applications and AIX-based technical applications. As the A/UX architecture evolves into PowerOpen, Apple will incorporate the latest Macintosh technologies into future A/UX products.

As part of the A/UX 3.0 release, Apple also announced version 1.1 of its A/UX Developer's Tools, which will be available from APDA this summer. This product includes Macintosh development tools that have been enhanced for the A/UX environment as well as standard-UNIX tools. The A/UX Developer's Tools version 1.1 product provides developers with the resources to create Macintosh, UNIX, and X Window System software as well as "hybrid" applications that incorporate Macintosh and UNIX functionality into a single environment.

A/UX 3.0 is only the latest in a variety of products we'll be releasing to expand Macintosh sales into the enterprise computing market, and you'll start reading more about those products here.

CD Highlights, June ’92

Welcome to *ROMin Holiday,* the June issue of the Developer CD Series.

International Market Guides and Apple International Contacts: These are country-specific guides that give you hints and directions on how to distribute and support your product globally. Also see Apple International Contacts for international third-party marketing and developer service contacts.

Tools & Apps: ADB Parser 5.0.5 allows for a thorough analysis of the Apple Desktop Bus (ADB). It provides multiwindow, modifiable device-naming and printing support. Also see the updated version of SCM Verifier 1.0a12. This collection of MPW scripts and tools reports on resource differences between “original” and localized files. An updated version of AppleGlot 1.1, another outstanding localization tool for the Macintosh, is included on this month’s disc.

C++ Tools: This month’s disc includes several C++ related tools. Take a look at the C++ Source Code Formatter. This MPW tool can be used to format C and C++ source code to a user-defined style. AFP C++ provides a C++ interface to the AppleTalk Filing Protocol (AFP) as documented in Chapter 13 of *Inside AppleTalk.*

ResEdit 2.1.1+fvew Editor: This is ResEdit plus a number of new templates and a Finder 'fvew' resource editor.

Human Interface Companion: The companion is an interactive supplement to the *Apple Human Interface Guidelines*. This preliminary version contains an introduction and 55 animated interface examples.

AE/AppleScript Course: Preview several representative modules from this course. This Developer University class will bring you up to speed on Apple events, the object model, and AppleScript.

VRAM & bit depth expInd.Pict: The PICT describes what bit depth is yielded by different VRAM configurations as well as current video cards.

Developer Notes: We've made several additions to the Developer Notes folder. You'll find the Developer Note for the new Macintosh Quadra 950. We've also included several older Developer Notes to use as references.

It Shipped!

Through the It Shipped! program, you can announce new and revised third-party products in *Apple Direct.* 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). Or contact Todd Luchette at (408) 974-1241 (voice) or (408) 974-3770 (fax). 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 May 1992.

Publisher	**Product (Version)**
U.S.	
Advanced A.I. Systems, Inc.	AAIS Full Control Prolog (3.0)
All-American Software Development Corp.	Health-Care Systems HyperTextBook (HTB) Level 1 (2.0)
	India HTBLevel 1 (2.0)
	Infectious Diseases HTB (2.0)
	Japan HTB Level 1 (2.0)
	Korea HTB Level 1 (2.0)
	Law HTB Level 1 (2.0)
	Mathematics History HTB Level 1 2.0)
	Music HTB Level 1 (2.0)
	Nervous System HTB Level 1 (2.0)
	Philosophy HTB Level 1 (2.0)
	Physics HTB Level 1 (2.0)
	Solar System HTB Level 1 (2.0)
	South Africa HTB Level 1 (2.0)
	Soviet Union HTB Level 1 (2.0)

	Space Exploration HTB Level 1 (2.0)
	Supreme Court HTB Level 1 (2.0)
	United Kingdom HTB Level 1 (2.0)
	United States HTB Level 1 (2.0)
	Yugoslavia HTB Level 1 (2.0)
Atex, Inc.	Atex Renaissance (1.0)
Autologic, Inc.	APS-SoftPIP™ (RA-03)
Brainchild Corporation	Brainchild Shortlist (1.0)
Emerson & Stern Assoc.,Inc.	Sound Bytes Developer's Kit (1.0)
Engage Communications, Inc.	EN+ (1.0)
Excel Software	MacAnalyst, MacDesigner (Educational Versions) (3.2/3.3)
	MacAnalyst/Expert (3.2)
	MacDesigner (3.3)
General Parametrics Corp.	Spectra*Star 450 (1992)
ISIS International	System 7 Pack (2.25)
LeadingTone Software, Inc.	Common-Practice (1.3)
Mainstay	MacFlow (3.7)
Maui Software	MacFORMation XL (1.0)
MicroMAT Computer Systems	MacEKG II (2.0)
MicroStrategy Incorporated	EISToolkit™ for Macintosh (1.1)
MikroLogix Software, Inc.	GradeKeeper (1.0)
ON Technology, Inc.	Meeting Maker (1.5)
Softsync, Inc.	Accountant, Inc. (3.0.1)
	Accountant, Inc./MultiUser (1.0)
	Expert Color Paint (1.0)
	ExpertWriter (1.0)
T/Maker	FaxMania (1.0)
Warner New Media	View From Earth, The (N/A)
Xplain Corporation	MacTutor Magazine (2.0)

CANADA

Folkstone Design Inc.	Anatomist: A Human Anatomy CD-ROM (2.1)

Image Club Graphics, Inc.	DigitArt Volume 24 Science & Medicine (1.0)
Image Club Graphics, Inc.	LetterPress CD ROM (2.0)
Spicer Corporation	IMAGEnation (1.01)
ENGLAND	
Neutral Ltd.	WarpDrive (1.0)
GERMANY	
ATEC	MacEnjoy (1.0)
JAPAN	
Digitech, Inc.	Speak English (1.0)

APDA Hot Product June ’92

Pick the debugger that’s right for your programming needs!

New! SourceBug version 1.0.1

An easy-to-use, source-level debugger for MPW-based programs that provides special support for debugging MacApp and object-oriented code. Allows you to debug your programs in the language in which they were written or in assembly language. NOTE: SourceBug is already included with E.T.O. For more advanced scripting features, use the SADE symbolic debugger. (See *APDA Tools Catalog.*)

APDA Part No. R0228LL/A $100

MacsBug version 6.2.2

The ideal assembly-level debugger for programmers who want power, expandability, and reliability when developing professional applications for the Macintosh. This version is compatible with the Macintosh Quadra. NOTE: MacsBug is already included with all MPW bundles and E.T.O.

APDA Part No. R0064LL/C $34.95

MacNosy version 2 and “The Debugger” universal version

“The Debugger” is a high-level symbolic debugger for the Macintosh that runs in a full multiwindow Macintosh environment. Features include high-level symbolic debugging of arbitrary programs, resources, and the ROM, source-level debugging for MPW version 3.0 or later compiled programs, and support for the direct debugging of THINK C version 3.0 or later project files.

APDA Part No. T0123LL/A $350

Developer University is now offering a special three-day Debugging Strategies and Techniques course that includes a working session with an Apple Developer Technical Support engineer. Contact the Developer University Registrar at (408) 974-6215 for Registration and Schedule information.

For ordering information, see “Now Available From Apple” in the News folder.

Now Available From Apple

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

Third-Party Products

Zortech C++ for the Macintosh version 2.1
T0438LL/B
$400.00

PICT Detective version 2.0.2b
T0125LL/B
$125.00

Macintosh Services Directory Summer/Fall
T0435LL/C
$14.95

APDA TOP TEN

1. E.T.O. Starter Kit
2. E.T.O. Subscription/Renewal
3. MPW C version 3.2 bundle
4. QuickTime Developer's Kit version 1.0
5. MacTCP version 1.1 Developer's Kit
6. Macintosh Common Lisp version 2.0
7. MPW C++ version 3.1
8. DAL version 1.3 VM/CMS Server
9. *Macintosh Programming Fundamentals* version 1.0.1
10. *Inside Macintosh,* Volumes I–VI and *X-Ref*

To place an APDA order from within the U.S., contact APDA at (800) 282-2732; in Canada, call (800) 637-0029. For those who need to call the U.S. APDA office from abroad, the number is (408) 562-3910. You may also send an AppleLink message to: APDA. If you're outside the U.S., 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*.

Designing for a Complex World

By Peter Deignan

When Mick Taylor quit the Rolling Stones in the late seventies, he left Mick Jagger and the boys with the tough job of finding another guitar for the self-proclaimed "Greatest Rock and Roll Band." For the next album, the Stones didn't see an obvious lead guitarist so they used a few of them. It's going to be just as tough to find a single person to fill Tog's shoes. Instead, you're going to hear a variety of voices. We may ultimately settle on one of them, or we might continue to rotate among a variety of human interface experts.

This month, the HI pen is in the hands of Peter Deignan, who runs the Human Interface Lab at Apple's IS&T (Information Systems and Technology) organization.

—Editor

Let me begin by making a confession: the Macintosh computer is no longer the simple, easy-to-use machine that it once was.

The main reason for this is that Macintosh users of 1992 are doing a lot more complex things with their computers than they did in 1984. While the original Macintosh was used mainly for simple drawing and word processing, the 1992 Macintosh does everything from CAD/CAM to remote database access. Even in areas such as word processing and graphics, users' expectations have grown exponentially. For a 1992 word processor to be taken seriously, it must have perhaps ten times the features of its 1984 ancestor. Of course, with this new power comes new complexity.

To maintain the Macintosh advantage, it's clear that we need to start incorporating new design techniques in our applications. In addition to the basic ten design principles from the *Apple Human Interface Guidelines* (metaphors from the real world, direct manipulation, see-and-point instead of remember-and-type, consistency, WYSIWYG, user control, feedback and dialog, forgiveness, perceived stability, and aesthetic integrity), I believe we need at least five more techniques when designing for the nineties:

- Constraints
- Intelligence

- Elegance
- Transparency
- Attention to detail

I have ideas about all of five of these, but for now, I'm going to talk more about the first technique, constraints, because of an experience I just had.

What got me thinking about the need for developers to design constraints into their applications was a recent market study conducted by a few Apple human interface people did to find out what people liked and didn't like about System 7.

After asking hordes of naive and not-so-naive users about it, they came up with lots of gripes ("takes too much memory," "printing is a lot slower"). But one result that amazed me was that many people thought it was much harder to use than System 6.

Upon further investigation, it turned out that this comment most frequently came from users who previously had not worked with MultiFinder. These people often found that, although it was great to be able to work with several applications at once, that same freedom often turned out to be a great source of confusion. Suddenly, they were forced to manage memory partitions and application layers, and they faced the possibility that if they missed a window they meant to click, their whole application "world" might change. No wonder that, like Russian pensioners facing capitalism, many of these users longed for the "good old days" when there just wasn't so much freedom.

Freedom Through Constraints: A Japanese Vending Machine

All of which leads me to a paradox: even in relatively modeless, free systems like the Macintosh, it's often good to constrain your users. The trick is intelligently limiting their options without taking away their ability to do the things they want to do.

I'm writing this article while working in Japan (having attempted to beat my deadline by dodging across the international dateline). As a bit of a soft-drink addict, I immediately noticed that the Japanese have an incredible number and variety of vending machines. Not only can you buy everything from canned coffee to magazines from them, but the drink machines will typically contain 30 different types of drinks, in an assortment of sizes.

Unfortunately, all the labels and directions on these machines are in a mixture of Katakana and Kanji, neither of which this *gaijin* stands a chance of reading. The amazing thing was that the designers of these machines used constraints so effectively that I not only got the large-sized diet cola I wanted, but I did so without ever committing an "error."

The machine I used had dozens of little compartments containing pictures of different roasts of coffee or brands of soda. Underneath each of these compartments was a little button with a light on it. Initially, all these lights are off, and you can press the buttons all day and nothing will happen. However, three other buttons, each labeled with a different amount of yen, flash at you, reminding you to put in money.

Once your money is in, the buttons underneath the pictures of the available drinks begin to flash. If the one you want is sold out, the button is dimmed. (This is like the way that buttons displayed by Macintosh applications are dimmed if you can't select them.) Again, if you push a button whose light is dimmed, no error occurs—but nothing happens.

After you select the kind of beverage you want, the button you selected lights steadily, and your attention is drawn to a final group of flashing buttons, over which are a small, medium, and large cup. Here also the machine's designer used constraints to let you choose without the possibility of error, for only the sizes that you've put in enough money to buy are lit. Since large diet colas cost ¥80, and I at first put in ¥50, only the buttons next to the smaller cups were lit. Needing the caffeine, I inserted ¥30 more, and the button under the large cup lit, allowing me to choose it.

As I drank my soft drink, I reflected on the fact that I had been subtly led by the machine through what could have been an extremely error-prone process. Because I was limited to picking only valid choices, I quickly got the drink I wanted without the machine's having to bleep at me and tell me I goofed. Compare this machine to American vending machines that, when they don't take your money without delivering the goods, bombard you with "item sold out," "put in more money," "exact change only," and other error messages. I can't speak for the quality of U.S. cars, but it seems that in vending machine design, the Japanese have us beat.

CONSTRAINTS CAN REDUCE COMPLEXITY

Like the designers of the Japanese soft-drink dispenser, when we write applications, we should try to take advantage of any natural constraints that exist. This strategy reduces the overall complexity of the user's task. For instance, a telecommunications program can let users choose whether they will connect by using a modem or over AppleTalk—but having chosen AppleTalk, they should not be asked to specify parameters that apply only to modems, like "phone number" or "baud rate."

Similarly, your applications should track what users are doing and not allow them to select options if they haven't filled in all the required fields. Note how AppleLink dims its Connect button until you have entered your name and password. This is far preferable to having it bleep, "You idiot, you forgot to enter your password!"

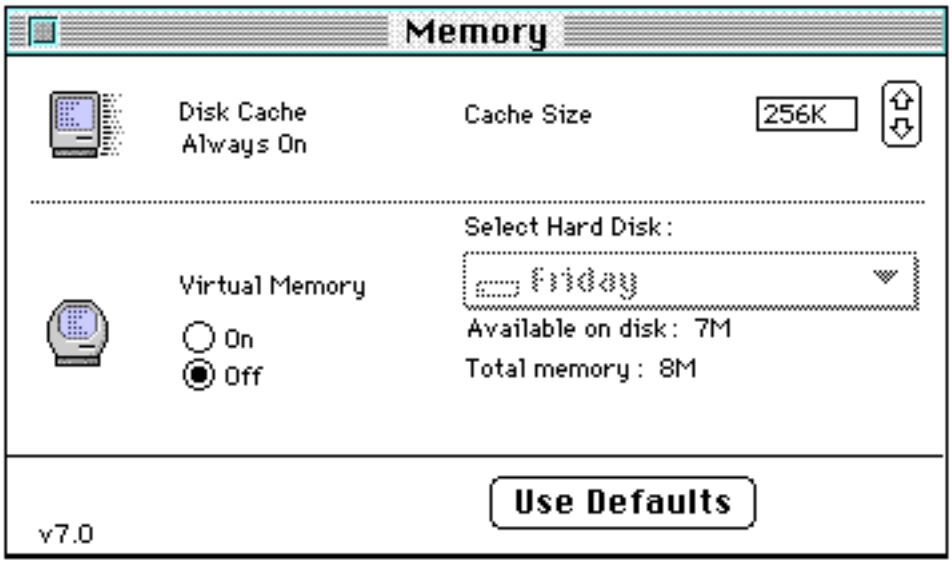
Moving From the General to the Specific

To make the best use of constraints, you should lead the user to first answer general questions ("Should I notify you when new mail arrives?"), that will eliminate the most possibilities, before moving on to specifics ("Should I beep, put up a notifier window, or both?"). In dialog boxes, for example, this means that you should either disable the specific items until the general items are selected, or arrange the items in the dialog box so that the user will tend to "read" the general questions first. Thus, if the dialog box text is in a Western language, the general questions should appear above, or to the left of, the more specific questions. (Most Western languages read from upper-left to lower-right; different rules apply in other cultures.)

A nice implementation of the general-to-specific rule is the virtual memory setup in the System 7 Memory control panel (see illustration). The control panel first asks whether the user wants to use virtual memory at all. If the user doesn't want to use virtual memory, he or she doesn't need to be asked what the size of the backing store should be, or what disk it should be located on. The control panel is carefully laid out so that if the user turns on virtual memory, the next item "read" is the pop-up menu for setting the disk to hold the virtual memory backing store. The amount of available space on that disk in turn constrains the size of the virtual memory store that can be created.

That's all for now (I find myself suddenly constrained by column space!), but I'm going to be putting a lot more thought into constraints and the four other "new techniques" of HI design for the nineties. If you have any thoughts on

these matters, and the complexity of System 7 application development, you can send me an AppleLink at APPLE.DIRECT.



Constraints in interface design: the System 7 Memory control panel with Virtual Memory turned off.

License Service API Promotes Acceptance of Network Licensing

Network licensing is seen as one of the most effective strategies for limiting unauthorized software use. (For more details, see “Antipiracy Technologies: Some Modest Proposals,” on page 1 of this issue.) Recently, the Software Publishers Association, along with such companies as Apple, Digital Equipment Corp., Hewlett-Packard, Lotus, Microsoft, Novell, and the Open Software Foundation (and about a dozen more), proposed a standard that may foster the use of network licensing.

The License Service API (application programming interface) provides an open, flexible way for software vendors to add network licensing support to their program in an implementation independent way. Currently, if a vendor wants to add network-license support to its product, it must add code that supports a specific network licensing technology. If a vendor uses the License Service API to support network licensing, the customer can use the vendor’s program with any network licensing technology that implements the License Service API.

The License Service API is open-ended and flexible enough to work with many network licensing technologies, computer platforms, operating systems, and networks. This API should encourage vendor acceptance of network licensing by giving vendors a single API that they can write to, knowing that the resulting program will work with a variety of network licensing technologies.

The first step is for vendors of network licensing technologies to support the License Service API in their products.

The next step is for software vendors to add network licensing support into their programs, using License Service API calls. For a copy of the License Service API specification, contact the Software Publishers Association at 1730 M Street NW, Suite 700, Washington DC 20036, (202) 452-1600.

GetNextEvent

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, Mail Stop 75-3B, Cupertino, CA 95014. For further information check the Events folder on AppleLink (path: 3rd Party Connection:Events).

June 23 through 25
PC Expo, New York, NY
Contact: H.A. Bruno, Inc.
(201) 569-8542

July 14 through 16
Mactivity'92
Santa Clara, CA
Contact: Winehouse Computer Company
AppleLink: Mactivity
(408) 354-2500

July 14 through 17
Object Expo - London
Contact: G.G. Schafran
(212) 274-0640

July 26 through 31
SIGGRAPH, Chicago, IL
Contact: Barbara Voss
(212) 752-0911

August 4 through 7
****MacWorld Expo**, Boston, MA
Contact: Mitch Hall & Assoc.
(617) 361-8000

September 8 through 12
Orbit
Basel, Switzerland
Contact: Kathrin Mäder
Switzerland 1 832 81 11
Fax: 1 830 63 06

September 23 through 30
Bureau 92,
Brussels, Belgium
Contact: BKK
Belgium 02 7627183
Fax: 02 7629434

September 29 through October 1
CAD CAM Show
De Hallen, Kortrijk, Belgium
Contact: De Hallen
Belgium 056 204000
Fax: 056 217930

September 30 through October 3
****SPA**
Washington, D.C.
Contact: The Conference Dept. of SPA
(202) 452-1600

Good Demos Sell Software

How to create a demo that sells

by Tom McLaren,
McLaren Associates, Inc.

In her monthly *ADWEEK* "Marketing Computers" column, Kristin Zhivago once said, "Buying software without a demo is like buying a car without taking a test drive. Most people want to get a feel for it." This is absolutely true, but I'll add a caveat: to sell software, the demo has to be a *good* one.

I have boxes full of demo disks from over the years, disks intended to sell software—demos that supposedly explain and convince and give a feel for the product. My guess: no more than a handful were successful. Many of them are poor examples of marketing communications. Haven't you sat through more than your share of bad demos? In this article you'll find out why a good demo is important and how to make sure your next demo sells software.

DEMOS DEFINED

This article focuses on demos that are used to sell a product—the pre-sales demo. However, there is plenty of overlap between this kind of demo and others, such as post-sales tutorial and help systems; much of what you'll read here could apply to most demos. In fact, any tool that helps people learn about your product will increase sales, since a product fully and successfully used gets good reviews and lots of word-of-mouth referrals.

The next time you create a sales demo disk, ask yourself how you want to use it. There are two ways to use demos as sales tools. A *direct response demo* is intended for a single viewer who doesn't have the benefit of someone to hold his or her hand or answer questions. The demo might be mailed to someone who has responded to an ad, or it might be used in a telemarketing campaign. A *point-of-sale* demo is a one-to-many people vehicle. It works well at trade shows and in stores.

Demos for Direct Response. A few types of demos that work well in the single-viewer environment are the simulation demo and the trial-size demo. A *simulation demo* (such as a self-driven "tour") can give users a feel for your application without allowing them to wander off-track and without requiring

lengthy setup and orientation. Also, since you "control" the demo you have a much better chance of ensuring that the customer feels some satisfaction after completing it. Indeed, a good simulation demo gives users a sort of self-substantiating proof: they'll believe the software works the way you claim it does; consistent prompting and feedback ensure that users understand how each feature works.

If your software generally demos well in-person ("live"), it will probably show well in a simulation demo. Simulation demos work best when the software accomplishes a concrete goal that gives the user instant results. For instance, "Do operation A, then do B, and voilà! Report C prints out—all the documentation your banker will need to process your loan." A simulation can be part of a point-of-sale demo, but since it requires a high level of involvement and interaction with the user, it works best as a direct-response tool. The user can sit down with the demo in his or her own environment and explore your product more fully. This, of course, would be more difficult to do in a retail environment.

If you anticipate difficulty simulating your software or if it is difficult to do a "canned" demo in person, your product probably isn't a good candidate for a simulation demo. Likewise, if your product can prove itself only when people use their own data, a trial-size demo may be your best choice.

A *trial-size demo* is used almost exclusively as a direct-response tool. This kind of demo uses a "quick start" book and a limited (or "crippled," to use a less appealing but more common term) version of the software. A trial-size demo must be accompanied by a compelling, clearly written book and bullet-proof sample data to make sure that people get the software up and running successfully. The last thing you want is a potential buyer trying your product, getting lost in the installation process or the application itself, and deciding that your product is too hard to understand—or takes too much time to use. Lose the customer there, and he or she will probably never come back for more.

Point-of-Sale Demos. The heart of a point-of-sale demo is the *attract loop*, a self-running presentation that draws passers-by to a computer and engages them in an interesting discussion of your company and product. Using compelling visuals and sound, this kind of demo helps the viewer associate an upbeat message with your product and company. Attract-loop demos work well in situations where salespeople need help.

For instance, at a trade show when your sales representatives are swamped, the attract loop can give browsers an overview of your company and product while they wait for your representative to talk to them.

GOOD DEMO, BAD DEMO

Making sure your demo is a good one is paramount to converting a customer's interest in your product into a sale. And a bad demo isn't only an ineffective waste of money; it can injure you. A customer who has a bad experience with the demo will be left with a very negative impression of your product, company, or both. Not only will you lose that sale, but probably others, as well. Remember, friends tell friends....

So it's important to take the time and effort to create a really good demo. Good demos are based on audience needs. Most of us have had it drilled into our heads: stress the benefits, not the features, of our products. (This is sometimes easier said than done. For instance: "Our printer is 50 percent faster." Is that a feature or a benefit to the user?)

Think of your demo as a story that must prove that your product meets your audience's needs. If you can't convince people of a product's benefits in a way that is meaningful to them, you probably won't make a sale.

Needs-based demos keep your audience's interest with a concise story and compelling conclusion. Likewise, a good demo is targeted at a very specific, well-defined audience. Of course, the demo may be useful to other people, but to be successful it must be focused on one particular audience.

In contrast, bad demos are usually based more on your love of the product's features and technology. They often are formless, overwhelming, and opaque, and are characterized by lengthy feature lists and no particular movement forward or conclusion. They try to reach everyone rather than a specific target, and end up communicating to no one.

Good demos are succinct, with a brisk pace appropriate to the subject. If you go into too much detail you'll lose your audience. Think of what's going on when you drift away from a book, a TV show or, heaven forbid, a magazine article. Good demos, like any communication, move quickly enough to hold interest, but linger long enough to cover essential information. Don't make your demo too long. For example, if the demo includes laborious company and product positioning followed by glowing descriptions of all the features, it's

probably going to lose the viewer, who is thinking, “Hey, get to the part that involves *me*!”

On the other hand, don’t make the demo too short. Sometimes a developer assumes that a list of features will convince me to buy the product—but I don’t even understand half the terms on the list. If your product requires it, give me enough education to get through the demo comfortably.

Good demos are thoughtfully designed—in terms of both graphics and user interface. A good demo always tells users where to look and what action to take. Bad demos confuse the user, and fail to distinguish clearly between demo messages and the software that is the subject of the demo. Good demos use the fundamentals of good visual communication to increase the power of the presentation. Bad demos use distracting graphic doodads that may be great whiz-bang attention-getters, but that will confuse the audience.

Good demos don’t require any special installation process. (The user should be able to pop the disk into the drive, double-click on its icon, and be off and running.) Bad demos, on the other hand, undermine customers’ confidence in their abilities to use your application.

Good demos are software- and user-tested. Use the same thorough, sound testing methods on your demos that you use on your products and your other sales/communications tools. Ask yourself such things as: Does the story work? Does the screen design work? How easy is it to install and use the demo? What message did it deliver? Did it reach the intended audience? This process is very important; bad demos reflect poorly on your design and development expertise.

In the end, you are actually the best person to decide what makes a good demo for your product. (For a synopsis of what makes a bad one, see “The Ten Most Deadly Demo Traits.”) My best advice: Call some 800 numbers, order some demos, and see for yourself what works and what doesn’t. Pick them apart, see why you like or don’t like them. Look at demos for competing products as well as new, unrelated products. Look at games, too. They’re great at getting and holding attention.

DEFINING AUDIENCE NEEDS

As I said earlier, a good demo is based on audience needs. And if you start the demo creation process by defining and specifying audience needs, it will be much easier to decide on content, pacing, and design. If you do a good job

defining audience/needs, the rest of the pieces should fall neatly into place. (I'll add that if you hammer out audience needs for your product communications programs—your marketing messages—as a whole, your advertising, PR, and demo messages will also fall into place.) In fact, both the look-and-feel and message of your demo should dovetail with all your communication pieces. Examine your software, logo, stationery, brochure, spec sheets, and ads—and company image overall—and be sure the demo fits.

The audience for your demo is a subset of the audience for the product. Even if your potential customer base is made up of various kinds of users (such as engineers, system administrators, and managers), each demo should focus on just one category.

How do you decide which group to focus on? First, ask yourself which group has the most purchasing power and which group will respond best to the demo. Sometimes the first and most obvious choice for your demo audience may not be the best.

For instance, we recently created a demo for a networking product. Even though system administrators had the purchasing power, we realized end-users were the primary influencers; Thus they were our audience. They were most likely to read our ads. And they were most open to getting the message in a simulation demo. For that matter, although a system administrator might not use it for a technical evaluation, the end-user demo would help him or her to explain the purchase and get buy-in from everyone else involved in the sale.

Next, create an audience profile that will help you design the demo. Be specific: What's their level of computer expertise? Their level of interest in this software category? Knowledge of competing products? Usual method for purchasing software? Typical hardware configuration?

Audience needs will determine which features you demo. For instance, what basic need does your product meet? Will it allow people to spend less time doing something tedious? Does it do something that results in better job performance? Whatever the answer, demo the product in a way that clearly shows the user how it will meet those needs.

Also, instead of thinking in terms of your *product* ("my product saves you time at the fax machine") think in terms of your *audience* ("you need to save time you spend on tedious, repetitive faxing"). Along the same lines, a demo that stresses that a product is fast and easy to use is not directly addressing customer needs. Even if the speed of your product is its selling point, you must craft the message

in terms of the specific needs of your audience. “Speeds up screen redraws” is not an effective message. “Spend less time waiting for your display” is a more appropriate message. A demo that clearly shows that the user will wait less time is more likely to close the sale.

Although having a product that is more convenient to use may be a need of some of your customers, the concept of convenience itself is too general and too relative to be the basis of a good demo. For that matter, just because your application is easier to use than other software products doesn’t necessarily meanthat using it is easier than doing the same thing manually.

Ideally, you should come up with needs that are so pronounced—so compelling—that if you can prove you meet them you will make a sale.

Needless to say, defining audience needs is an important, painstaking process that may take awhile. Everyone, from your engineers to your warehouse manager, has strong ideas about the product and the audience. But once you decide on a primary audience and its most pressing needs, you will be able to create a successful demo.

LET THE PRODUCT PROVE ITSELF

Once you’ve settled on the primary audience and its needs, here’s how to translate that into a story for the demo. First, examine the needs list and determine exactly how to prove that the product meets these needs. Then double-check the steps needed to perform an operation. You may find that some features or processes just don’t demo well. It is acceptable to skip a few of the non essential steps, but avoid misrepresenting how basic operations work.

Next, create a story based on a normal project or chain of events that features your product in the starring role. Think about how your primary audience would approach this project and then create a scenario that proves that your product can do the job.

You may want to provide a “step-by-step” tutorial. These can be extremely effective, but they have a down side as well. On the up side, some people need to feel they are really giving a product a workout.

A step-by-step tutorial accomplishes this by taking users through the actual menu choices and keystrokes required to perform a key task with your product. Also, a tutorial is very effective in helping users believe the product really works the way you say it does. And, the more interactive the demo, the more your prospect is involved in the sales process. On the down side, your audience may

not wish to do such a rigorous evaluation, and a tutorial may be a bit too detailed for them. (You can see how understanding audience needs and targeting the right people is critical.)

DEMO VERSATILITY

Demos are versatile tools. If you plan well, you can use them in a variety of ways. They aren't useful only for the near-term; demos can also work with long-term marketing programs. For instance, if you're rolling out several new products in the next two years, consider creating a modular demo that can be easily updated.

With minor modifications, demos can be used in some very creative ways. I urge you to get the most out of your investment by thinking about all the ways you might use demos.

For instance, if you're increasing co-op advertising with resellers, think about how to tie a demo to a special promotion—such as contest disks. In a contest disk, users are given an incentive to go through essential parts of the demo.

One way might be to reward them for matching three graphics; that is, at the beginning of the demo they are given a graphic. If they find two other graphics during the demo that exactly match it, they can return the disk to the developer and receive a prize.

Also, direct-response demos are used not only in conjunction with ads ("call this 800 number for a free demo") or direct-mail campaigns, but also as handouts at trade shows and as leave-behinds for salespeople. They can be bundled with other products, and can even be bound into magazines.

Demos can also be customized to feature bundling or comarketing arrangements. For example, a label company customized the demo of a leading database application to showcase the section about how to run labels. They bundled an offer for the demo inside their label packages.

Also, point-of-sale demos can be used as props (and prompters) for salespeople. By replacing text with bullet points, the demo becomes a salesperson's portable presentation.

PROJECT MANAGEMENT AND TIMING

A demo is a unique communications medium—a hybrid of software, print, and video. A successful project leader should be able to focus on the customer, be familiar with software development cycles, and manage review cycles involving

both marketing and technical people. And most important: the project leader needs to nurture the underlying communications goal and keep it intact throughout the process.

Manage the creation of your demo as you would a software development project. Demo development works best when you follow a plan through prototype, interim versions, and tested golden master. And make sure to leave time for both software testing and user testing. Early on, your tests should determine if consumer action goals are being met and if the interface works properly. Later in the process, testing should focus on software functionality and compatibility with all target hardware.

Think long-term. If you design a modular demo, it will be easier to upgrade next year. If you might eventually localize it for use in other countries, think about what you can do now to facilitate that process when the time comes.

To estimate development time, consider these ranges and the size of your project. During the R&D stage, you learn a development system, build tools for your job, or both. This may take a couple of weeks or even up to several months. For scripting and prototyping allow a minimum of a week; four weeks or more is probable. For building a first full version, allow 2 to 8 weeks. Each iteration (new version based on review feedback) takes anywhere from 2 to 8 weeks.

The total development time (excluding R&D) is anywhere from 5 to 20 weeks. The low number is very ambitious and the high number not-too-high for a large job.

However, sometimes you're under pressure to get it to market and must do whatever you can to shorten the cycle. If you must compress it, do so carefully. If you "throw together" a demo, it will be clear to the audience that this is what you did. You should carefully weigh the possible consequences of getting an ineffective demo to market quickly against taking a little more time to produce a demo that will sell software.

It's important to approach demo project development as seriously as you would any software project; but design it using communications goals, such as you would for a brochure or other communications piece. If you concentrate on the audience and its needs—and avoid the ten most deadly demo traits—you'll end up with a demo will help sell software.

Tom McLaren is the president of McLaren Associates, Inc., a Portland, Oregon company that specializes in creating software demos for sales and user training.

****

The Ten Most Deadly Demo Traits

#1: Talking to several audiences at the same time. (“I think they started to lose me when they spent all that time talking about the output speed to 3-D rendering devices....”)

#2: A long-winded features-based story. (“Gee, I’m only on screen 5 of 9—maybe if they supplied a glossary I’d actually understand some of what they’re talking about....”)

#3: Not being clear about where to look or what do next. (“Perhaps it’s just me, but I can’t tell if I should click here, or watch the flashing dog icon, or just wait for something to happen....”)

#4: Too long. (“Silly me, here I am thinking I can learn about mailing-label software in under an hour....”)

#5: Too many choices of what to do next. (“Hmmm...should I choose, Benefits, or maybe Quick Tour, or maybe Easy Reports, or perhaps Executive Summary? Or maybe I’ll come back to this when I have more time....”)

#6: Cluttered windows. (“This is exactly what I’m looking for in an application, something even more unappealing than my desk on its worst day....”)

#7: Condescending, idiotic sample data. (“Hoo-boy, this is truly real-world, an inventory example featuring the short-people’s stilt company....”)

#8: Difficult installation. (“Well, I’m ready right now to order software from a company whose demo I can’t even install....”)

#9: Carrying simulation to a fault. (“I’m tired of typing in phone numbers—or is this actually a typing proficiency test....?”)

#10: Offering unclear info about hardware requirements. (“Jeez, Marge, the demo shut us down. I guess our system just isn’t good enough for their demo....”)

Marketing to User Groups

Creating a Program That Pays For Itself

by Terry Fleming,
Timeworks, Inc.

"You should consider marketing to user groups." You've probably heard it before (over and over again). Indeed, there have been numerous pitches urging developers to do so. If you aren't convinced or you don't know how to get started, this article is for you. I'll explain why Timeworks made the investment in a user group program, how it pays for itself, pitfalls we've faced, and some tips about getting started.

Why a User Group Program? For the last two years, Timeworks has devoted money, time, and other resources to this activity—for several reasons.

Every survey ever done indicates that people base purchase decisions primarily on word-of-mouth referrals. A user group is a collection of people who enjoy using computers and helping others to use them. They enjoy sharing information with each other (and with you, if you ask). By gaining their support, you get the mind share of the largest collection of purchase influencers in the industry.

The flip side is that you could receive a negative reaction if user group members don't like what they see. But in my experience, most groups will be so darned glad you took a personal interest in them that they'll love you to pieces. (That translates to: they'll be more likely to purchase your product or service).

There's another bonus to working with user groups: the average group contains a sampling of all the various hardware and software combinations imaginable. Members cover the entire spectrum of skill levels and groups are run, by-and-large, by volunteers. Not only is this a "buffet" of willing beta test candidates who can tie your product in knots, but also many of these users have a very broad feel for what the market wants.

They are very outspoken and will gladly tell you how they perceive your product, company, policies, your new necktie—anything you might want to know (and some things you might not).

A Program That Pays For Itself

Influencing future purchases is a long-term goal of a developer's user group program. In many cases, developers may hesitate to launch a program because it requires an investment that results in a seemingly intangible, unmeasurable long-term payback. However, our program gives us the long-term benefits but also pays for itself immediately *in a tangible way.*

Timeworks offsets the cost of its program by selling products directly to user groups, an approach formulated when we started the program two years ago. We analyzed the fixed costs (travel, meals, equipment rental, and so forth) to visit a user group in person, and decided to offset that by selling products at the meetings.

But beware: there are a host of caveats that accompany this strategy. First, user groups are all-too-aware of current retail channel prices—especially mail order prices. If you decide to sell direct, you *must* beat the best price they can get anywhere else, or don't even bother.

After all, you are approaching these people because they are special customers. Everything you do should reinforce that. If you can't (or would rather not) offer a discount, so be it. But at least don't offer your product at $10 higher than the going street price. This has a tendency to irritate group members. If you're going to offer a special price, make sure that it *is* special.

Avoiding Channel Conflict. Adjusting your selling price for the user group seems simple enough, but what happens when the local dealer gets wind of it? You're selling to the group for less than the dealer can buy your product, a valid concern on his or her part.

At Timeworks we continue to be sensitive to this potential channel conflict, although we have not experienced a problem. When we present a product to a user group, we generally offer it at about 50 percent off the suggested retail price. We diminish potential channel conflict by restricting the special price to that presentation only. Also, just as in direct-mail marketing, many members will patronize their favorite dealer to buy the product anyway, which is fine. Our primary goal is to build awareness while providing a low-cost opportunity for user group members to try our product.

This way, we get both the ongoing, more intangible return on investment (influencing future purchasing decisions and getting users to tell their friends) *and* an immediate, tangible, and measurable payback that helps defer the cost of the user group program.

GETTING STARTED

To help you decide if marketing to user groups is right for you, here are some things to consider—and some of the pitfalls you'll face—when launching a user group program.

Make the Commitment. In our experience, to realize a return on your investment you must take user groups seriously; allocate your resources according to the level of commitment you desire and to a level that *you can support.* For example, it's easy to make a user group mailing with the best intentions and make a host of promises you *intend* to fulfill. Resist! (Remember the old saying about the road to hell being paved....) Take it slowly and increase your efforts with a degree of control. Think of it this way: if you open a restaurant that can seat 100 people and you invite 1200 to come, when more than 100 customers show up it doesn't matter how good the food is.

Choose the Right People. To succeed, you need a person (or staff) devoted solely to the necessary activities. Timeworks has devoted several full-time people and the resources needed to support them to work solely with user groups.

Furthermore, the most effective people for the position are those who can play multiple roles. Such people must be able to simultaneously position the product in the minds of user group members as "something they need" (marketing director), handle questions about the features and benefits they'll realize by owning your product and close the sale (sales manager), fluently compare and contrast the competition at any level (technical guru), and convey the enthusiasm of a child at Christmas (little kid). User groups have a variety of expectations that this person (and your program) must fill. (For more information, see "Meeting User Group Expectations.")

Support this individual (or individuals) with administrative back-ups who are organized and who follow through well, and you'll be able to make a major impact. Try to do all of these things with existing, non-dedicated resources on a part-time basis and you'll also make a major impact—the wrong kind.

Allocate Adequate Resources. Allocate the proper equipment and processes needed to keep track of user groups and all the related information.

Managing the data is an ongoing task. Group presidents/ contacts change from year to year (as do their addresses and phone numbers). So do factors such as group size, focus, number of special interest groups change, and so forth.

We use a custom-designed data base that automates the process of organizing and tracking more than 1000 international user groups. While you won't need anything this elaborate to get started, bear in mind future needs as well as today's while doing your planning. You'll quickly accumulate data once you start.

Plan for Demo Equipment. When planning your resources and budget, don't forget demo equipment. Have you heard that most user groups will provide equipment for demos—free of charge—and all you have to do is show up? This is often true, but it isn't always a benefit. For example, the equipment requirements to show a database will be totally different from those needed to demonstrate a 24-bit color image-processing program. If you want to show a QuickTime movie and all they have is an old black-and-white overhead display panel, you'll need to bring (or rent) your own equipment. In this case, factor the equipment cost into the overall cost of visiting the group, plug it into your cost/benefit formula, and see if meeting with this group still makes sense for you. If it doesn't, it may be better to avoid visiting this group for now.

Use outside resources as much as possible. One excellent potential source of help is Apple. The Apple User Group Connection is a group that is dedicated to working with Apple's user groups and various professional associations. It maintains a database of hundreds of groups across the country who participate in the program.

For more information about how to take advantage of the variety of user group opportunities Apple makes available to developers (and how to use the mailing list), contact the Developer Support Center at (408) 974-4897.

[Editor's Note: *For general information, you can also send an AppleLink message to USER. GROUPS, or to American Online address: Apple UGC. In addition, Apple has published Developer Connect, a newsletter full of information, resources, and recommendations about successfully working with user groups.*

To request a copy, contact the User Group Connection at the electronic mail addresses given above.]

Select Target Groups, Design Programs Accordingly. Groups come in a variety of flavors. They have horizontal interests (community groups, for example) and vertical interests (corporate, education, government, and so forth). Some are very small, some are very large.

We learned early on that it was not economically feasible to visit all smaller groups in person. So we developed the Timeworks' Ambassador Program, which maintains relationships with more than 1000 Macintosh and DOS user groups. We enlist a volunteer at each small group to be our ongoing contact. We send that person a copy of our product (which will belong to the group), literature, and support needed for him or her to make a presentation about it, and order forms to give to group members (who send orders directly to Timeworks). As an incentive for the ambassador, we're testing a new approach: After we receive the first order from a group member, the ambassador receives a free personal copy of our product.

The Timeworks' Ambassador Program, has changed as we learn from our experiences. One thing stays the same, however: The program entails a tremendous amount of administrative time and overhead.

We do visit the larger groups personally, because they represent the greater opportunity. (We reach more people per presentation and have full control over how the information is presented.) However, they also present the most problems. First, scheduling. Every developer wants to visit them, so you may have to schedule with them months in advance. (For more information about scheduling with user groups, see "The Scheduling Nightmare" below.)

These arrangements can be especially important if you are launching a new product; you'll want to schedule as many user group visits as possible during the first few months after the intro. If you wait too long, user group members will have already seen and heard about your product, and your presentation won't seem as "special."

Second, the equipment necessary to demo your product to 500 people is usually more sophisticated and demanding than that needed to present to a group of 50.

Finally, while big groups are more cost effective to visit in most cases, I'll add a caveat to what I said earlier about visiting small groups: don't underestimate the sales potential of reaching them. We did one presentation for a local community user group; one member was the director of facilities and systems of

a major national fast-food chain. As a result of our presentation, this person became a very valuable corporate contact. You never know who you'll run into when you go to a user group meeting.

Questions to Answer. In short, if you can answer these questions you're probably ready to go: Do you understand your cost/benefit ratio? Is it at an acceptable level? Will you be doing in-person presentations or establishing some type of remote "liaison" program such as Timeworks' Ambassador Program, or both? Will you sell your product directly to the group? At what price? Are you willing to give away product to create excitement at a presentation and help seed your market?

TWO YEARS LATER

At Timeworks, after two years of putting the pieces in place, our user group program is ready for a major expansion. Our user group department is both a marketing/sales vehicle and a successful profit center. Was it worth the investment? Absolutely. We can now launch a new or updated product directly—and affordably—to the most influential segment of our customer base.

Can you afford to market to user groups? Can you afford not to?

Terry Fleming is the director of user group services for Timeworks, Inc., creator of the Publish It! desktop publishing products, based in Northbrook, Illinois.

****

The Scheduling Nightmare

One of the more time-consuming administrative tasks is a nightmare called "scheduling." Getting onto a user group's agenda isn't as easy as it sounds.

Here's why: Most user groups meet once a month. Some meet even less when a regular meeting day falls on a holiday. Some don't meet during the summer since meeting attendance can drop dramatically, due to vacations and an occasional conflicting tee time. There are only 12 "second-Wednesdays-of-the-month " in a year, so you only have 12 opportunities to visit a given group each year. Add to this the fact that the "second Wednesday" may be the

meeting day for several important groups you want to visit, and the scheduling difficulty increases.

Finally, the more established, large, and influential the group, the more likely it will book speakers 8 to 10 months in advance. Commit to the group's first available date; it's worth it.

Since most user groups meet in the evening, you can optimize the use of the needed travel dollars, too. You can combine any other day-business you have in a given city with an evening appointment with a user group. In most cases, you'll have to arrange the user group meeting first, and then back-fill your other business around it.

Also, when you're trying to schedule a presentation you'll sometimes find that you'll have to share the "spotlight" with another speaker. Don't let that intimidate or stop you—it can be a blessing in disguise. Sharing with someone else only serves to broaden the overall appeal of the meeting and will draw more people to it. It costs you just as much to talk to 50 people as it does to talk to 150, so look at this as an opportunity instead of a curse. (A major exception would be when you're asked to speak alongside a direct competitor. In this case, politely offer the entire spotlight to your competitor because that's just the kind of person you are.)

****

Meeting User Group Expectations

What do user groups expect from a developer? They expect fairness, free product, honesty, free product, inside information, free product. Did I mention free product? Don't get me wrong; they don't demand free product, but they do expect it—much in the same way that people expect you to dress up for a wedding. You really don't have to, but it'sconsidered disrespectful if you don't.

Be prepared to donate at least one unit of your product to a group. Obviously, if you're demonstrating a $20,000 color printer, of course you probably cannot afford to donate one; but if you're showing a $79 software package, be prepared to give away more than one. The group will raffle your generous donation at the end of the meeting, and maybe one winner will review it for the

group's newsletter. If the winner won't be receiving it, consider donating yet another copy to the group for review purposes.

User groups also expect you to give a mediocre presentation. What??! I didn't say they wanted one, just that they've come to expect it. So one way to really distinguish your product and company is to deliver a dynamite presentation. Here are some helpful do's and don'ts for making your pitch:

•*Don't give them an "annual report" on your company.* When you present to a user group, the only person in the room who cares about your company's fiscal status or marketing position is you. Leave it out of your presentation. Let your competitors make this mistake and bore the audience to tears.

•*Don't take user groups lightly.* These people are on the cutting edge of the industry. Individually they may be a notch or two below your best techno-experts, but as a group they possesmore depth of knowledge than you might imagine. Not only are they are knowledgeable about your product, but they also often know your competition, and how your product compares, as well or better than you do—not in the same marketing/positioning sense as you'd like them to have, but in the real-world "is-it-really-better" sense. If you try to buffalo them, they'll eat you alive.

You have three options: Know what you're talking about; send someone who does; or at least be humble enough to admit your limitations to them before getting too far out on a limb. Besides, it's far better for you to humble yourself than for them to humble you—which they can. Therefore,

•*Do know your product inside and out.* User groups expect this, and you can't pull the wool over their eyes. Many presenters seem to know more about their companies than about the products they present, so you'll be well served to understand the basics about your product. While you'll be forgiven if you don't know about some of its more complex capabilities, if you don't know whether it is compatible with a group member's favorite word processor (for example), you can kiss your credibility goodbye. Anticipate that audience members have seen your literature and have read the current reviews (why else would they invite you to speak?). Read them yourself and be prepared to discuss anything and everything—good and bad—that you've read.

•*Don't show them products they can't buy right now (or at least very soon).* User groups expect to see the newest, hottest things, but resist the temptation to show them something that won't be available for ten more months. Sure,

they’ll gobble it up, but they’ll also be disappointed when they have to wait to buy and use it.

However, if you want to solicit input about a new or upcoming product, go ahead and knock their socks off with it (but make its availability—or lack thereof—clear). Otherwise, leave it in the car.

1992 Worldwide Developers Conference Tapes Still Available

Missed the Worldwide Developer's Conference (WWDC)? Need a refresher on some of the vast amount of material that was presented there? We have good news for you—you can still purchase audiotapes and videotapes of most of the 90-plus WWDC sessions.

At the May conference, Apple gave the official word on such new technologies as AppleScript, O.C.E., RISC, and Apple's new imaging architecture as well as development tools, programming tips and techniques, enterprise computing, and marketing. For a list of available tapes, or to place an order, contact Mobiltape via AppleLink (MOBILTAPE) or by phone at (805) 295-0504. Audio tapes cost $8 each and video tapes are $29.95 (plus an additional shipping charge for both).

Apple and Macworld To Cosponsor Mexico Developer Seminar

Apple Computer, Inc. and MACWORLD magazine will jointly sponsor the Mexico Developer Seminar, to be held July 23 to 24, 1992, at the Hotel Nikko Mexico, in Mexico City.

This two-day seminar will provide U.S. developers with information on and contacts within the Mexican computer market, one of the fastest-growing markets for the Macintosh in Latin America.

To obtain more information, or to register for the conference, contact Sandy Butler, Events Manager, Mexico Developer Seminar, MACWORLD Communications, Inc., 501 Second Street, Suite 500, San Francisco, CA 94107, USA. Phone: (415) 267-1749. Fax: (415) 442-0766. AppleLink: BUTLER.S.

APPLE DIRECT

Apple Direct is Apple's monthly developer newspaper, covering business and technical issues for decision makers at development companies. It is published by Apple Computer, Inc.'s Developer Support Systems and Communications (DSSC) group.

EDITOR:
Paul Dreyfus (AppleLink: DREYFUS.P)

TECHNICAL WRITER/EDITOR:
Gregg Williams (GREGGW)

BUSINESS & MARKETING EDITOR:
Dee Kiamy (KIAMY)

PUBLICATIONS AREA MANAGER:
Hartley G. Lesser (H.LESSER)

PRODUCTION EDITOR:
Lisa Ferdinandsen (LISAFERD)

CONTRIBUTORS:
Juan Bettaglio, Cindi Cain, Pat Calderhead, Beth Dawson, Suzanne Dills, Marian Djurovich, Sharon Flowers, Lynda Lucero, Monica Meffert, Stacy Moore, Silvio Orsino, Katherine Parsons, Jessa Vartanian, Ana Wilczynski

MANAGER, DSSC:
David A. Krathwohl

CONTENT GROUP MANAGER:
Greg Joswiak

FILM:
Aptos Post, Aptos, CA

PREPRESS:
Prepress Assembly,
San Francisco, CA

PRINTER:
Wolfer Printing Co., Inc.
Los Angeles, CA