Apple direct

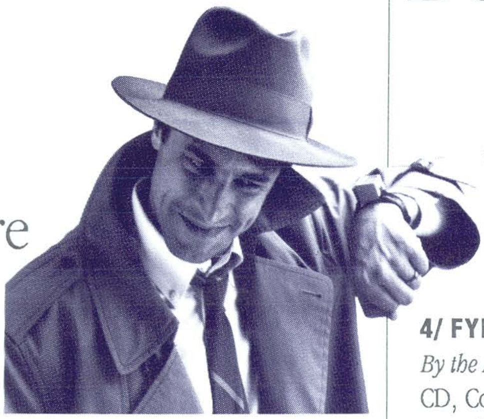
VOL. 2, NO. 9

AN INFORMATION RESOURCE FROM APPLE DEVELOPER SERVICES • JUNE 1990

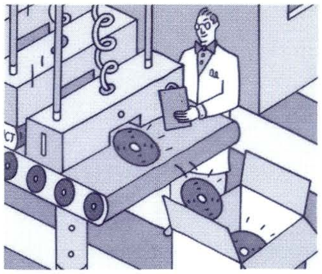
Looking into the Future

June 1990

Features



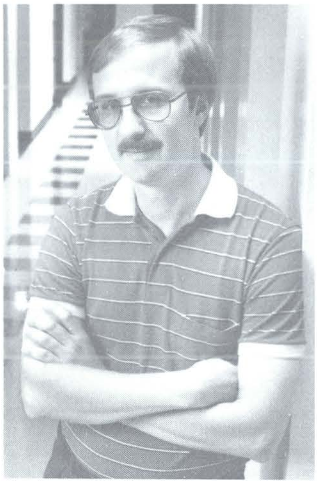
Departments



®
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010
TLX 171-576

Editor's Notes

ISDN—Not Just Another Petty Phrase



ACRONYMS ARE THE UNWELCOME BAGGAGE OF A WORKER'S vocabulary, but they *are* useful when they squeeze a big idea into a small word. ISDN, which stands for Integrated Services Digital Network, is such a word. Its goal is to create a worldwide network of high-speed data/video/speech lines as useful—and ubiquitous—as today's network of telephones. Because ISDN uses the same physical connections that telephones do, it will eventually be available just about everywhere that phones are today—and because it is an international standard, it will be available all over the world.

ISDN includes both a *digital network* (hardware) and *integrated services* (software that will make it useful). ISDN is being packaged in two sizes. The minimum configuration, called the *Basic Rate Interface* (BRI), contains two digital lines (called Bearer, or B, channels) and a third Delta (or D) channel; the combination is sometimes abbreviated as "2B+D." Each B channel, which can be used for any kind of data, can transmit and receive at 64,000 bits per second, whereas the BRI D channel, which helps control the B channels, can transmit at 16,000 bps. If this isn't enough, there is a "bigger" channel, called the PRI, for Primary Rate Interface. It specifies one PRI D channel at 64,000 bps and either 23 B channels (in North America and Japan) or 30 B channels (in Europe) at 64,000 bps each. Talk about bandwidth!

The Macintosh ISDN Developer Toolkit. Last month Apple announced its ISDN Developer Toolkit, which is Apple's first step in supporting ISDN technology. Apple wants to seed selected developers with the Toolkit to get them started on creating innovative ISDN applications.

The ISDN Developer Toolkit contains two things. One is an ISDN NuBus™ card, which provides the hardware and software necessary to allow the Macintosh to connect to an ISDN network. The other is a new Toolbox manager, called the IVD (Integrated Voice/Data) Manager, that adds new tools to the Macintosh Communications Toolbox. The IVD Tool lets your program control various telephone functions; the ISDN Serial Connection Tool provides serial communications over ISDN data channels.

ISDN is an exciting technology because its speed and its network of connections will make entirely new kinds of software and services possible. The extensive reach of the ISDN network means that you will take for granted the ability to move text, graphics, sound, and images to any location you choose—and the implication of that is that *it will no longer matter where your data (or computing power) is stored*. And *that* will certainly change computing.

If you're seriously interested in doing ISDN development, call the Developer Services Hotline at (408) 974-4897, and ask about the seeding requirements for the ISDN Developer Toolkit. Someday we will take global access to data for granted, and ISDN will certainly be one of the vehicles that will make that possible.

Gregg Williams

Gregg Williams, Technical Editor

Apple*direct*

PUBLICATIONS TEAM

EDITOR
LISA RALEIGH

TECHNICAL WRITER/
EDITOR
GREGG WILLIAMS

DEPARTMENTS EDITOR
BILL FREAIS

PRODUCTION
COORDINATOR
DIANE WILCOX

DESIGN / PRODUCTION
FISHER & DAY

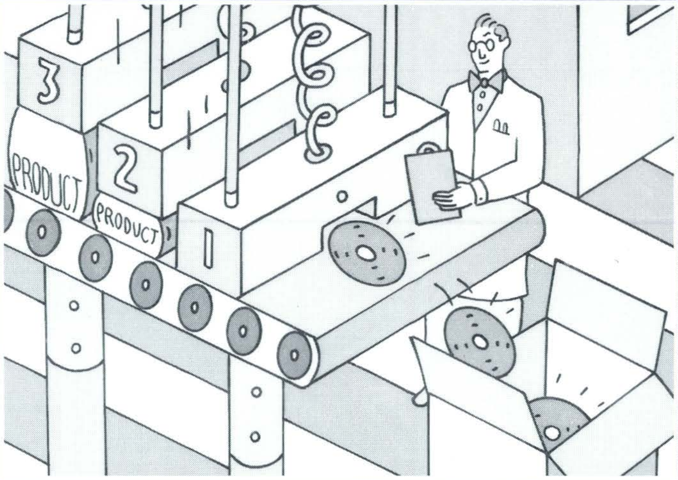
MANAGER,
DEVELOPER PRESS
DAVID KRATHWOHL

Apple Direct is published monthly by Apple Computer Inc.'s Developer Press group, part of the Developer Services organization.

MENTION OF PRODUCTS IN THIS newsletter is for informational purposes only and constitutes neither an endorsement or recommendation. All product specifications and descriptions were supplied by the respective vendor or supplier. Apple assumes no responsibility with regard to the selection, performance, or use of the products listed in this newsletter. All understandings, agreements, or warranties take place directly between the vendors and prospective users.

LIMITATION OF LIABILITY: APPLE makes no warranties with respect to the contents of products listed in this newsletter, or the completeness or accuracy of this publication. Apple specifically disclaims all warranties, express or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. *Cover photo by John Lund. Editor's photo by Ron Grant Tussy. Cover story illustrations by Craig Hansen. Other illustrations by Steve Turk.*

FYI



Make Sure the Field CDs Your Product

APPLE CONTINUES TO OFFER YOU THE OPPORTUNITY TO PUT a demo of your product into the hands of the Apple® field sales force and authorized Apple resellers.

We recently released version 4.0 of the Apple Reference & Presentations Library CD to more than 3,000 Apple field sales and marketing personnel and more than 2,000 authorized Apple resellers.

The Library CD contains more than 650 megabytes of information on Apple products, Apple service and support programs, Apple literature, and developer and Apple demos. The field and resellers use this CD as a centralized cross-market resource.

The next version of the Apple Reference & Presentations Library CD (version 5.0) will be released in early fall of this year, and the producers are now on the lookout for more demos to include. If you would like more information, contact David Grabel (GRABEL1) via AppleLink.® Otherwise, send your demos to The Apple Reference & Presentations Library CD, c/o David Grabel 20330 Stevens Creek Blvd., Mailstop 36-B, Cupertino, CA 95014.

MacApp 2.0 Goes Final

FOR QUITE SOME TIME you have been hearing Apple nudge you in the direction of object-oriented programming and the use of MacApp®. And now MacApp 2.0 has gone final—no more 2.0ß5 or 2.0ß9, but 2.0 final. And it is available from APDA™ today.

MacApp has undergone many changes since its last final release, version 1.1.1, covering the class library, tools, documentation, and samples.

The MacApp class library has been enhanced with a view hierarchy that makes it much easier to create and work with dialogs and windows in your applications. And, in this latest release, MacApp has been enhanced with new list management methods (procedures).

MacApp 2.0 includes ViewEdit, a direct-manipulation graphical view builder, that is used in constructing windows and dialogs, and configuring buttons, fields, and other user-interface objects in them. New to this release of MacApp is Mouser, a source code browser, used for looking at and editing MacApp and application source code by class hierarchy rather than file-by-file.

Documentation for MacApp 2.0 includes an *Introduction to MacApp 2.0 and Object-Oriented Programming*, (also available separately from APDA) which introduces the concepts of OOP and how to get started with MacApp. The *MacApp 2.0 Tutorial* leads you through the step-by-step creation of a simple MacApp program. How-to recipes are included in the *MacApp 2.0 Cookbook*. And the *MacApp 2.0 General Reference* explains the MacApp architecture and theory of operations. A new online *MacApp 2.0 Class and Method Reference* documents every field, method, and global in MacApp 2.0.

MacApp 2.0 gives you a head start with six sample programs, from a simple MacApp application with 70 lines of code, to a complete spreadsheet-style application. Two of these samples are written in C++.

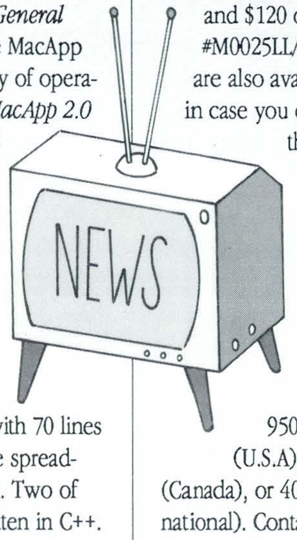
Although written completely in Object Pascal, MacApp-based applications can be written with multiple object oriented languages, including Object Pascal, C++, and Object Modula. MPW Object Pascal and MPW C++ are available from APDA. Object Modula is available from the MacApp Developer's Association.

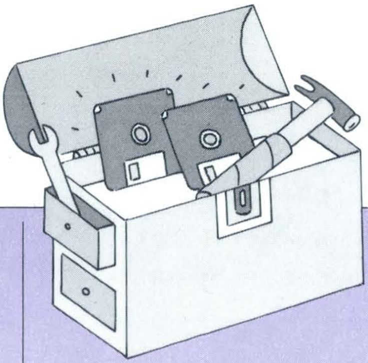
As mentioned in *Apple Direct* last month, MacApp can be compiled with THINK Pascal 3.0, and the MacApp 2.0 package from APDA contains a disk from Symantec, Inc. including the Converter program for modifying the MacApp sources to work with their compiler.

The complete MacApp 2.0 package including all MacApp 2.0 software, tools, and online-documentation, on both floppies and a CD-ROM, and complete printed documentation (the four manuals described above) is available for $275 (APDA # M7022/D).

For previous customers, updates including the *MacApp 2.0 General Reference* and the release notes, along with complete software, are available for $80 on CD-ROM (APDA # M0742LL/A), and $120 on floppy (APDA #M0025LL/C). The manuals are also available separately in case you didn't purchase them when you first purchased MacApp.

For more information, contact APDA, 20525 Mariani Ave., MS 33-G, Cupertino, CA 95014; 800-282-2732 (U.S.A), 800-637-0029 (Canada), or 408-562-3910 (International). Contact MADA (the MacApp Developer's Association) at P.O. Box 23, Everett, WA 98206; (206) 252-6946.





Mac Tech Notes Stack

A HYPERCARD STACK CONtaining Macintosh Technical Notes, 1985–1989, is now available from APDA™ as part# M0215LL/B. For more information or to order the stack, contact APDA at (800) 282-2732 in the U.S., or write to APDA, Apple Computer, Inc., 20525 Mariani Ave., M/S 33-G, Cupertino, CA 95014-6299. You can also Link: APDA.

Communications Toolbox News

THE FOLLOWING COMmunications Tools for the Macintosh Communications Toolbox are now final (version 1.0). They're available through APDA and Developer Tools Express℠ and are ready to be licensed for commercial redistribution. (For information on licensing, contact Apple Software Licensing at [408] 974-4667; AppleLink: SW.LICENSE). These tools provide applications with data-connection, terminal-emulation, and file-transfer functionality:

TTY (Teletype)
Terminal Tool
Text File Transfer Tool
XMODEM File Transfer Tool
Serial Connection Tool
Apple Modem
Connection Tool

Tools in the following list are now Beta and will also be available through APDA (although they cannot be licensed for commercial redistribution yet).

VT102 Terminal Tool
VT320 Terminal Tool
AppleTalk® ADSP
Connection Tool

There's also a new Communications Toolbox folder on AppleLink's Developer Services Bulletin Board. You can use this folder to get the latest toolbox information from Apple, to share toolbox information with other developers, to download tools for testing purposes, and to share tips and techniques.

Apple/Digital Update

ON MAY 1, APPLE AND Digital Equipment Corporation announced DEC LanWORKS for the Macintosh, a product set that furthers Apple's commitment to its corporate customers by providing new solutions for user connectivity. DEC LanWORKS for the Macintosh includes the following Apple and DEC™ components:

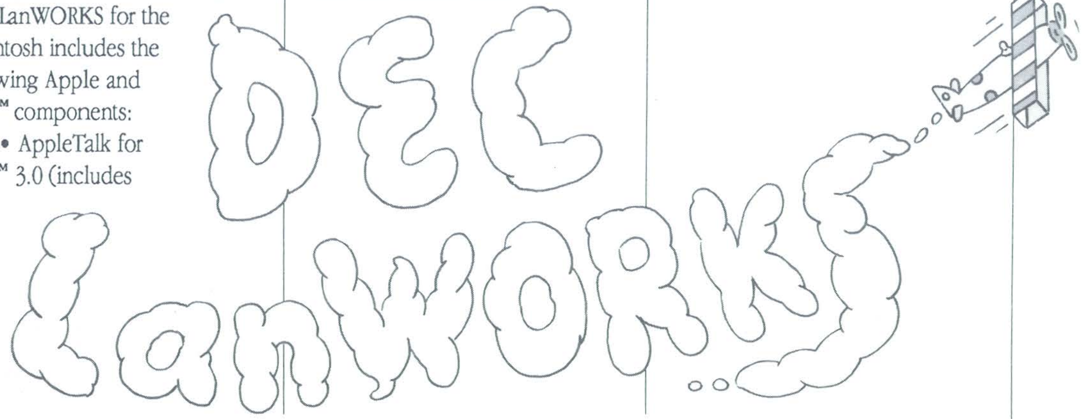
- AppleTalk for VMS™ 3.0 (includes AppleTalk internet-routing capability over DECnet™ networks).
- AppleTalk-to-DECnet Gateway.
- DECnet transport (free option to the basic package implemented as a Macintosh Communications Toolbox tool).
- Apple's AppleTalk Filing Protocol (AFP) compliant file server, based on the enhanced Alisa technology.
- Apple's Printer Access Protocol (PAP) compliant print services, based on the enhanced Alisa technology.
- MacWrite/MacDraw/MacPaint to/from the DDIF (DEC's document interchange format) document converter.
- Data Access Language database server and client for access to relational databases on a VAX.
- MacTerminal® 3.0 VT320 emulator with (Local Area Talk) support via the Communications Toolbox.
- MacX, DECwindows (X Window) display server.
- Electronic-mail interface to VAX/VMS Mail.

Beginning in mid-July, DEC will sell and support DEC LanWORKS for the Macintosh through its worldwide organization and reseller channels. DEC has also announced plans to offer a Macintosh client for ALL-IN-1 by late 1990 and a bundled MicroVAX-based LAN server. DEC will sell an Ethernet card for the Macintosh as well. Apple intends to develop an AppleMail-to-VAXmail gateway and a connection with DEC's electronic-mail systems via X.400.

DEC LanWORKS for the Macintosh is one result of a two-year development effort that has yielded the foundation for industrywide compatible, consistent, and supportable connectivity between AppleTalk and DECnet (DEC's wide-area network or WAN) environments.

For Apple's third-party Macintosh developers who offer products for the business market, DEC LanWORKS provides a stable, endorsed development platform.

With VAX™ connectivity, your products will be able to reach out beyond local-area networks to access the information and resources on which the business community relies. And they'll be able to do so with little additional work on your part—the VAX services are based on AppleTalk. For more information, Link Apple's Digital Integration Evangelist at EVANGELISM.



New technologies present many possibilities that you can start planning for now

LOOKING INTO THE FUTURE

There is no such thing as *the* future—but if there are *ten* futures for tomorrow, there will be a million next year, maybe even a billion a few years down the road. How can we possibly make plans in the face of that? The answer: We pick the future we like the best and then aim for it. In the case of future Macintosh products, it might look something like this:

Dr. Janet Hall is listening to a presentation on outpatient therapy when her Assistant, a personal office the size of an average hardback book, beeps twice. She leaves the lecture, finds a chair in the hallway outside, and opens the Assistant. A cartoonlike face appears in a corner of the screen.

"Your majesty," the Assistant, Parrie, says. (Janet smiles at the customized greeting; sometimes it's the only humor she gets in an all-too-serious day.) "Dr. Aldrich needs to speak with you."

"Put her on."

Dr. Marcie Aldrich's face appears in another window. "Janet, Parrie showed me how busy you are today, so I'll keep this short. I just got the X-rays back on a patient, and there's something bothering me that I can't pin down. Can you take a look at them?"

"Sure," Janet says. The screen fills with an X-ray image with four white diamond shapes; Marcie's face retreats to one corner of the screen. Janet points to each of the diamonds in turn, listening to the heartbeat sounds that Marcie has captured at those points.

GRAMS - VOICE I/O

By Gregg Williams, *Apple Direct* staff

"Parrie, enlarge this area here, and enhance it." She looks at the result. "That doesn't tell us anything. Do we have anything else?"

"The office database has a new imaging module from Medical Visions. Shall I install it in your viewing tool?"

"Yes, then run it on the same area." A thermometer scale appears on screen, fills, and vanishes; the enhanced image appears. "Parrie, check the national cardiac database for anything like this one."

"Working..." Parrie says. "No matches."

"OK, consider your last two actions as one task. Now do it to the areas here, here, and here."

"Working...." Janet scans the patient's history in another window while Parrie works. "There are three matches on image 4," Parrie says.

"Marcie, look at this. I think we've found a weakening in the myocardium. For a 30-year-old male, that's not good. I'd recommend some further tests—a B5 and a B30."

"That sounds right," Marcie says. "Thanks for the help—lunch is on me next time."

"How sweet of you," Janet says, "especially since my Assistant just billed your office for a consult. See you soon."

As any card-carrying enthusiast of science fiction knows, as a *specific* prediction of the future, the above scene isn't worth the paper it's printed on. But it does indicate a good direction for future Macintosh software (future in this context meaning two to five years from now)—and you need to take your first steps now if you want to get there eventually.

Mix 'n' Match Programs. Just as Dr. Hall can add an imaging module (or "engine") into her viewing tool (what we would call a program today), users will soon be able to mix and match program modules to create a program that works the way they want it to. Users will be able to—for example—mix a text-display engine from company A, a spelling checker from B, a grammar coach from C, and a search facility from D, thus obtaining a word processor that they *really* like. It also gives them one spelling checker that works—the same way, and with the same dictionary—with *any* program they use.

System 7.0's Inter-Application Communication and AppleEvents™ (a suite of standard messages built with IAC) give you the framework you need to create modular software. The good news for you, the developer, is that you can now write smaller products that are simpler to design, implement, debug, and maintain. Of course, this also means that you will have to learn a new way of programming—one that concentrates on doing one thing well and uses other modular programs to do the work that your program does not know how to do.

What this means to you, now: To cash in on the IAC-related benefits of System 7.0, you need to implement both the Core and Required AppleEvents that Apple has defined. These events allow anybody's program to call on any other program—including yours—for certain basic services (such as opening a file and printing a document). Then you should create your own private AppleEvents and "wire" them to each module of your program so that other programs can call on your program to do work for them.

User Programming. *User programming* is the term for any set of actions (inside a program) that the user can specify and later cause to happen. In the scenario above, Dr. Hall's instruction to "consider your last two actions as one task [and do it three times]" constitutes user programming. User programming has been with us for many years; the "macro languages" of Excel and most other spreadsheet programs, the HyperTalk® language in HyperCard,® and "keyboard macro programs" such as QuicKeys™ are good examples.

Under System 7.0, you don't have to create your own scripting environment—all you have to do is to make your program AppleEvents-responsive (as described above). Then when Apple and/or some third-party company offers a *scripting environment* (a place where a user-created list of commands, or *script,* generates a sequence of AppleEvents that do the user's bidding), your program will be ready to participate fully in the process of user programming.

Eventually most (or all) Macintosh programs will respond to a rich set of AppleEvents and will be controllable via scripts. When this happens, the Macintosh will take a quantum leap in usability as *system-wide scripting*—the ability to devise and execute scripts that can cause the Macintosh to do almost anything the user can do manually—becomes possible. Users can be much more productive by creating scripts that automate the sometimes-tedious process of doing the same thing over and over again on a personal computer. Preparing for user programming is an important way of multiplying your program's usefulness.

What this means to you, now. Examine your programs to see if there are some possibilities for adding private AppleEvents that allow your program to perform useful work when other programs request it. Apple Developer Technical Support will help you and other developers share information on your custom AppleEvents.

Integrated Media, a Daily Resource. According to Apple Fellow Allan Alcorn, the conventional idea of multimedia—sound, video, and graphics, all on a laserdisc—is too limited, because it is a many-to-one, passive-user experience, so expensive to create that only *companies* can afford to do so. Integrated media on the Apple Macintosh will be different: It will be personal, active, one-to-one, and a part of daily life—just as the graphics/sound x-ray image is in our example. (Alcorn likens integrated media capabilities to styled text. Before the Macintosh, styled text was expensive; only companies used it, and they used it for special occasions—annual reports, for example. Because the original Macintosh included styled text as an inherent capability of the computer itself, now *everybody* uses styled text, and they use it every day, for everything—even for a memo to someone else in your group.)

Apple plans to make support for new media an integral part of the Macintosh Operating System. At the Worldwide Developers Confer-

Technologies of the Near Future

Not everything in the future is in the far future. Here are some items that you should think about—they are all possible now or with the upcoming System 7.0:

- *Background processing under MultiFinder® (or Finder™ 7.0).* Background processing is the ability for deactivated programs (i.e., all programs except the current one, which is running in the top window) to "steal" little slices of time for noncritical tasks that don't require user intervention. (In System 6.0.x, these features are in MultiFinder; starting in System 7.0, they are all in the Finder, and there is no longer a MultiFinder file.) Two example uses of background processing are background file printing (used by Macintosh system software) and background uploads and downloads (used by telecommunications programs to allow the Macintosh to do other useful work while sending and receiving files).

You must write some additional code to make background processing happen, but the result—a program that still gets work done, even when it's not the active one—is worth it. For more information, consult the *Programmer's Guide to MultiFinder* (item M7044 from APDA, phone [800] 282-2732) and Chapter 26, "Process Management," of the draft version of *Inside Macintosh Volume VI* on the System 7.0 developer CD-ROM you recently received. (The path to use is "Big Bang: Inside Macintosh: Inside Macintosh Volume VI: Viewer Version: 26 Process Management".)

- *The Edition Manager.* This new manager, also part of System 7.0, allows you to "publish" and "subscribe to" parts of documents; when one part (or "edition") changes, other documents containing that edition are automatically updated. Make sure your program can do both of these.
- *Direct Manipulation for Fonts:* Many users don't really care what size the text is; they just want to make it "this high" or "this wide"—so these users should have the option of direct text manipulation in addition to the usual manipulation by point size. Why can't we just select the text and pull one corner of the selection rectangle until the type is the size we want it? Here's something else: If you have a piece of a graphic that includes text, wouldn't it be nice to resize the graphic object and find the text resized as an object rather than as a bit map? The ability to extend direct manipulation to typography and type manipulation is an extremely attractive feature waiting to be implemented in System 7.0 applications.
- *Collaborative Applications.* This refers to a category of software that promises to do for a small group of people what the personal computer has already done for the individual—make them more productive. It can be software that allows people to work on a common task (either in real time or asynchronously), or it can help people communicate better with each other. There have already been a handful of attempts—mostly good tries but nothing really successful. Two points to keep in mind: First, be a group worker before trying to create collaborative applications; and second, people want software that's simple and easy to use—don't provide tons of features they won't use. (Also, see Alex Knight's Viewpoint, "Have Your Computer Call My Computer (and We'll Do Lunch...)," in the May 1990 issue of *Apple Direct.*

ence last May, Apple announced QuickTime, an extension to the Macintosh OS that will enable applications to control media peripherals and their data. For more details, see "The Lowdown on QuickTime" on page 13 of this issue.

The medium most likely to be added first to documents is sound. Why? Macintosh already has a lot of support for sound, and it's a medium that most of the Macintosh installed base has the horsepower to use. The Sound Manager has been enhanced for System 7.0, and it's software that *any* Macintosh can use. (However, two capabilities—the ability to play continuous sound from the hard disk while doing other work and the ability to play multiple channels of sampled sound simultaneously—are available only on the Macintosh SE/30 and Portable and all Macintosh II models.)

One part of the Sound Manager is a set of sound routines called MACE (Macintosh Audio Compression and Expansion). These routines provide 3:1 and 6:1 sound-compression and decompression routines. Since this is done in software only, the MACE routines will work in any Macintosh Plus or later model. (Parrie's voice output is one of the most noticeable aspects of the story above, and that's quite feasible to do today, using MACE.)

S. Joy Mountford, head of the Human-Interface Group within ATG (Apple's Advanced Technology Group), says that the emphasis at Apple will be to provide system-level tools for text, sound, animation, and graphics so that you, the developer, can create products that integrate one or more of these into a "document." Imagine creating a video presentation that uses video, text, sound, animation, voice, and graphics—and then selecting Print to Video from the File menu and ending up with a videocassette containing the same presentation!

What this means to you, now. You should add "hooks" for future data types into your next program. (You should also implement the Core and Required AppleEvents and your own private AppleEvents.) You might not be able to implement, say, animation in the first version of your word processor, *but* by the time version 2.0 comes out, animation may be the hottest, flashiest thing around—and you will want to add it, whether doing so is simple or not. But even animation—if you keep it simple—is possible on all the Macintosh computers in use today.

The first thing you should add to documents is sound—QuickMail™ and Microsoft Mail, for example, allow the user to send voice messages, and more products will follow. You should consider adding voice annotation—the process of adding to a document voice messages that play back when the user activates them.

Acknowledgments

Numerous people contributed ideas to this article. They include Allan Alcorn and Alan Kay (Apple Fellows); David C. Smith and S. Joy Mountford (ATG); Darin Adler and Bob Glass (Product Development); Steve Goldberg, Jim Stoneham, and Michael Wallace (Product Marketing); and Glen Fiance (Evangelism).

The Future According to Szetela

At last May's Worldwide Developer Conference, David Szetela (manager of Developer Services at Apple) gave a talk on present and future technologies (with examples) called "Application Directions for the Future." Audio and video tapes (90WWDC-33 and 90WWDC-133, respectively) of this lecture are available from Mobiltape Corp.; contact them at (805) 295-0504 for more details.

"Open the Pod-Bay Doors, HAL." Despite the industry's failure to get a computer to understand unrestricted speech, voice input and output *will* be part of the future—and unlike Parrie, they don't have to be or "look" intelligent to be useful. Already, numerous companies have produced partial solutions that are great successes —speech output is much easier to do than speech input, but the latter is possible if you limit the speaker's vocabulary, require each word to be spoken separately, and/or "train" the computer to the user's voice. (For an example of current speech technology, see page 21 for this month's Developer Outlook column, "Voice and System 7.0," by Tim Morgan.)

System 7.0 makes it much easier for your program to use voice input, because it allows you *not* to reinvent the wheel—in this case, the "wheel" is the whole hardware/software combination that makes voice input happen.

Now somebody has to invent this, but it doesn't have to be you. As long as your program responds to the private AppleEvent messages you define for it, input can come from a voice input subsystem—or anything else—that can send those messages to your program.

What this means to you, now. Look for occasions to use speech—but only if voice I/O

Agents implement what you might call "semi-intelligent, semi-independent assistance."

makes the program easier to use, not harder. You should also design your upcoming programs so they can do their jobs through private AppleEvents.

Farallon Computing has released a low-cost microphone/software package called the MacRecorder Voice Digitizer (it's cheaper than their existing MacRecorder product); this digitizer is meant to be used with programs that can store voice input. Also, Farallon has just introduced MediaTracks, which allows you to create a "tape" of whatever's happening on the screen and annotate it with both graphics and sound. Contact Farallon (415-596-9000) for details.

AY
E'S
HE
UB
SLUGGO, WE NEED HELP GETTING TOGETHER A LIST OF PHONE NUMBERS!
OK BOSS!
WE'LL SORT 'EM TOO!
M
TO
WHAT TH-

Agents: Personal Gofers. No less than Apple Fellow Bill Atkinson has called them "slaves without guilt." David C. Smith (formerly of Xerox, now in Apple's ATG) put it this way: "Agents are inevitable technically—plus they fulfill our deep-seated desire to have a staff"; he went on to call agents "the icon-level idea for the 1990s." Whether they have a humanlike shape or not, agents bring a new metaphor to the average user: that of a servant or representative who can go and do things in your place, reporting back only when it has finished.

In the story at the beginning of the article, the cartoon figure, Parrie, is an agent, and it's obvious that she (forget sexism—*Star Trek* taught us that all humans respond better to a female voice) is a very visible part of the computer's interface. Some people believe that the agent should not be a humanlike image, citing potential problems such as users' having unrealistic expectations of an agent that looks and acts human, or the irritation of being repeatedly interrupted by a "dumb" agent. Still, the allure of anthropomorphic agents is overpowering—they provide a metaphor that anybody can relate to.

Agents implement what you might call "semi-intelligent, semi-independent assistance" —software that understands how to do a task without your guidance and reports back to you when it is finished. In many cases, an agent's work is to go for things—search external databases, for example, for information that meets the criteria you've set for it. One example of agent software that exists now (and *isn't* anthropomorphic) is the Macintosh Navigator, a user interface and terminal "front end" for CompuServe. Once you to set up the search criteria, Navigator will connect with CompuServe at a set time and download all the new material that meets the criteria. You might also want an agent to look at everybody's networked appointment books and automatically schedule a meeting when everyone involved is free.

Agents go well with user programming. One straightforward way of telling an agent what to do is to write a script that explains how to get the work done (existing examples include several telecommunications packages and the custom scripts you can write to control them). Once you have given the script to the agent, it can, at some specified time, interpret and execute the script's statements to do the work for you.

What this means to you, now. Start thinking of tasks that agent software can handle and how you will implement them. If you are intimidated by the prospect of creating a visible cartoonlike agent, you aren't required to do one—just open a text window to talk to users—but give them options so that the agent can be made to work the way they want it to (for example, they should be able to specify how frequently the agent can interrupt their work). Also, start thinking about how your agent software can be more effective by "learning" its user's behavior.

The Agent as Animator. Agents also add a very important dimension that is largely missing now in personal computers: independence. Computer programs today are really pretty boring—in some ways, no better than a pile of rocks. You can stack rocks in different ways; you can pick one up, drop it, and watch it fall to the floor. But you don't expect rocks to fly figure eights in the air or build little rock houses for themselves. Personal-computer programs are very much like this. In most cases, the state of your computer every morning is exactly the same as it was the night before, and the programs don't do anything until you personally initiate an action. Why should that enormous amount of computing power go to waste?

The agent can be used as a metaphor for an assistant that can work on your task both while you are using the computer *and while you are gone*. The agent provides a framework in which this work can take place and a user interface that most people can easily understand. Eventually, this will lead to *distributed computing*—programs that use idle computers on the network to get their work done.

Take the simple example of an agent that inspects your electronic mail and prioritizes it according to some criteria you've given it. Our first attempts at doing this will probably be inefficient, so let's say it takes the agent a monstrous *five minutes* to analyze each letter. If you get, say, around 25 pieces of mail a day, you can't afford to let a program do that while you wait for it during the day (let's say it uses too many resources to do it in the background). This task would take about two hours on its own, which may be a lot of time in the day but is only *one-eighth* of the 16 hours a day that the computer's idle. This gives you some indication of the magnitude of problem that can be solved during your computer's idle time.

(People are already using the Macintosh's idle time. For example, ON Technology's ON Location indexes your new documents in the background, while you're not using your Macintosh; if you return, ON Location stops its work until you stop using your Macintosh. Similarly, ALSoft's DiskExpress II works on optimizing your disk whenever your Macintosh is idle. Even though you wouldn't think of these programs as having agents, they do...if you want to see it that way. Remember, the concept of *agent* is a metaphor.)

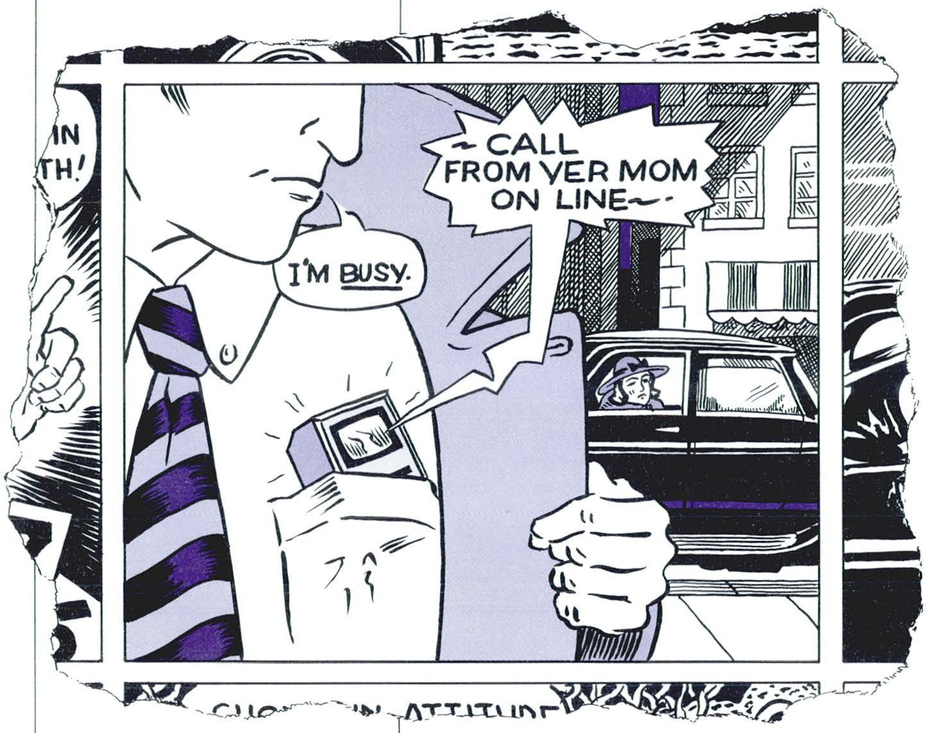
What this means to you, now. How many computer problems can you think of that you haven't attacked because they'll take up too much time? The agent will be most popular when it is doing time-consuming, routine, medium- to low-priority jobs that don't require your intervention. Make your program use idle time; also, make it work in the background under MultiFinder (see the "Technologies of the Near Future" sidebar for more details). You can probably find something useful to implement now, and the IAC in System 7.0 will make even more things possible and easier to do.

You Are What You Have Access To. One of the subtler points of the opening story is that Dr. Hall has access to a lot of data and that the location of the data—any computer that her book-sized Assistant can reach by telephone—does not matter. Her Assistant may not itself have more than, say, 100 megabytes of mass storage, but it can get access to gigabytes, maybe even terabytes, of information from remote databases—without Dr. Hall's knowing or even caring about where the data comes from. (It's also possible for the application being run and/or the computing power itself to be on remote computers.) The idea of transparent remote access (to data, programs, and computing power) will be a very important feature of future computers, because it will liberate them from the limitations dictated by the configuration of the computer itself.

What this means to you, now. There's certainly nothing to stop you from creating a program that transparently accesses a remote

Most computers are unused 16 or more hours out of every 24. Why should that enormous amount of computing power go to waste?

database by phone—many programs already do this. Even if you don't think access to remote data is a big issue for your program, you should still implement the high-level calls in the Data Access Manager; they're easy to implement, and they make it possible for your program to access remote databases without additional work on your part (for more details, see my article "Simplifying Remote Data Access," page 6, in the October 1989 *Apple Direct*). You should also do all your communication with the outside world via the Communications Toolbox.



Eventually, we may "wear" computers like we now do a watch or a breast-pocket notepad.

It is also very important that your program be a "team player" in this future where applications call on each other to get work done. This means that you must work well with AppleShare® and that you implement the Core and Required AppleEvents and your own private AppleEvents.

Portables 'R' Us. Eventually, computers will become an essential part of our daily lives, and we may "wear" them like we now do a watch or a breast-pocket notepad. Cellular-telephone or some other technology will allow these computers unfettered access to remote computers. That communication technology will be an essential component of the transparent remote access mentioned above.

What this means to you, now. You don't have to wait for a book-sized Cray to enter the world of portable computing. Several kinds of portable computers already exist, and their users are eager to buy truly useful software and hardware for them. Don't know what kinds of programs they'd like? Try using one in the field to find out what connectivity they need to the Macintosh world. Also, use a Macintosh Portable in the same way to find out what kinds of Portable-related products are needed.

Prepare for the future by asking yourself, "What kinds of remote data might be useful to our users? What does our company eventually want to give them?" These goals on the horizon will help clarify your first steps in this direction.

Less Is More. As users' connected hard disks daisy-chain out to the horizon, users will be begging for software that doesn't show them everything they have, only the things they might want to see right now. An electronic-mail-sorting agent is one good example. ON Location is a good existing package that delivers a much needed capability: *information retrieval by content, not label.*

Another thing you can do to make less (data) into more (information) is to use time to help present the data. Imagine 50 images from a stop-action animation of a flower bud opening. Viewing them as 50 images, you would have to do a lot of comparisons to discern any patterns in the growth process. On the other hand, show them as about two seconds of video (then show them again slower, faster, and so on), and you can see the same patterns —and more—instantly. Animations do this all the time, as do simulations that show some kind of real-time display. The human eye is very good at seeing image sequences and visual patterns. By using these talents, you can show collections of data that are more complex than would otherwise be possible.

What this means to you, now. The field of retrieval by content, not label, is an important one, and there are many ways to do it—you may find a product opportunity here. With existing technology, you can create agents (you may be more comfortable calling them "filters") that sort or prioritize collections of items for you.

The Future, Today. Because products usually take more than a year to conceive, design, implement, test, and release, you can't afford to

Continued on page 24

THE LOWDOWN ON QUICKTIME

...and other components of Apple's media-integration strategy

By Lisa Raleigh, *Apple Direct* staff

In his speech at the Worldwide Developers Conference last May, Apple CEO John Sculley talked a great deal about multimedia—specifically about how Apple is moving a range of multimedia projects from research to product status. Sculley also spoke about a cross-platform media-interchange standard, for which Apple is inviting developer proposals, then turned the podium over to Vice President Don Casey (the leader of Apple's new multimedia engineering group) to fill in the details.

Casey outlined Apple's media integration strategy, the core of which is QuickTime, an extension to the Macintosh Operating System that will manage the control of media devices and multiple streams of media information (see illustration, next page). What QuickDraw has done for two-dimensional graphics, QuickTime promises to do for handling of sound, animation, and other media data types. Working with developers, Apple plans to develop the QuickTime spec and API (application programming interface) by the end of this calendar year, Casey said.

"Developer success is critical to our success and we will strive to be as open as possible with developers. We will work with developers to set an industry standard multimedia document architecture and we are open to selectively licensing technology from developers," said Casey.

Cutting to the Quick. QuickTime will include core technology that will help developers deal with the problems inherent in integrating new media types across all application categories. The idea is to achieve a cross-platform/multi-vendor media interchange standard. QuickTime will include:

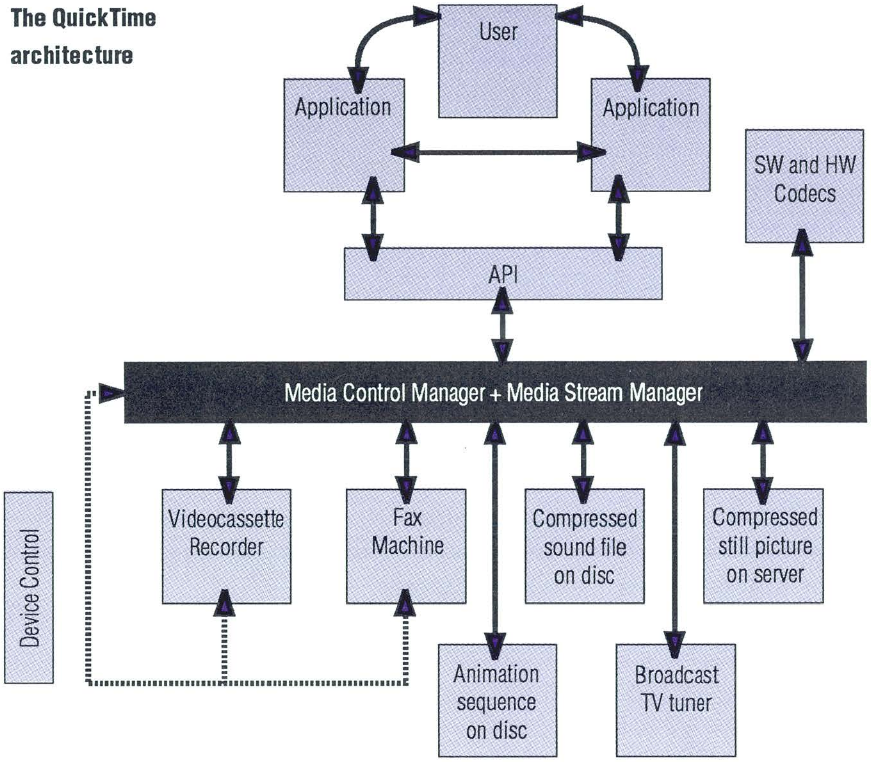
- an environment that solves the problems of dealing with real-time data such as sound and video;
- arbitration between multiple data streams (i.e. sound and video);
- a standard device control and data interface to all applications;
- a media-compatible document architecture; and
- the development of human interface guidelines for integrating new types of media.

To define QuickTime, Apple is calling for all developers to submit their ideas. What should Apple be doing in the multimedia arena, and how should QuickTime evolve to meet your needs? Apple is taking feedback on QuickTime via the new Media Integration Discussion folder on AppleLink. (AppleLink path: Developer Services Bulletin Board: Developers Ask Each Other: Media Integration Discussion). Apple is also interested in investigating technology for possible incorporation into QuickTime. Look in an upcoming issue of *Apple Direct* for more information about how to submit technology/co-development proposals to Apple.

Apple is calling for all developers to submit their ideas about QuickTime.

Scalable Capabilities. Besides QuickTime, Casey also described Apple's intentions in terms of hardware development. Rather than focusing Apple's efforts on providing a single hardware platform configuration or a "multimedia machine," Apple intends to provide developers with a scalable set of hardware and software capabilities that are integrated across the modular Macintosh product line. Sound, video, CD-ROM and HyperCard are the key technologies that Apple will focus on. While most innovations will take place first in the modular family of Macintosh systems, many of these capabilities will also eventually migrate to the compact line:

The QuickTime architecture


User
Application
Application
SW and HW Codecs
API
Media Control Manager + Media Stream Manager
Device Control
Videocassette Recorder
Fax Machine
Compressed sound file on disc
Compressed still picture on server
Animation sequence on disc
Broadcast TV tuner

Sound. Casey announced that sound input and output will be incorporated in all future modular Macintosh systems. He also announced that Apple will provide a text-to-speech replacement for MacinTalk™ that will produce higher quality output with substantially improved accuracy and intelligibility. It will also support multiple voices and languages. Apple will also be establishing human-interface guidelines for sound input and management.

Video. Apple intends to develop real-time software compression/decompression technology that will run on today's modular Macintosh systems—with a reduction of as much as 90 percent in the storage required for digitized video. Apple also plans to integrate composite video output into future modular Macintosh computers and in NuBus display cards. This will enable Macintosh video output to be sent directly to any video projection or VCR device.

CD-ROM. With CD-ROM, you can expect to see Apple drive the cost down and increase performance, creating products for a broader base of customers. At the same time, Apple has efforts underway to make CD-ROM easy to integrate into Apple CPUs.

HyperCard. In case you missed last month's *Apple Direct*, Apple is continuing to evolve HyperCard in a big way, the latest evidence of which is HyperCard 2.0 (see the May 1990 *Apple Direct* cover story for all the details).

Apple has already found HyperCard to be an excellent development environment for multimedia applications and will continue to evolve it as a "staging area" for new multimedia capabilities. For instance, Apple is planning HyperCard extensions for multimedia support, such as the control of videocassette decks.

The announcement of QuickTime and the public description of various multimedia activities are the latest steps toward encouraging developers to join us in media integration. In next month's *Apple Direct*, we'll tell you more about why you might want to look at multimedia for your own products—whatever they might be—and how you might go about integrating new media types.

In a Nutshell

Here are the announcements you missed if you weren't in attendance at the Worldwide Developers' Conference. Apple intends to:

- Deliver the QuickTime specification and API to developers by the end of the year;
- Include sound I/O as a standard feature of all future modular Macintosh computers;
- This summer, provide human interface guidelines for new media types;
- Deliver improved text-to-speech technology within one year; and
- Deliver software-based video compression and decompression within one year as an APDA product.

THE FINE ART OF FINE-TUNING

A HyperCard 2.0 example illuminates several aspects of human-interface design

By Gregg Williams

What does it mean, now, to be writing software professionally? Years ago, expectations were lower and you were doing quite well if you had printed documentation, a plastic bag to put it in, and a program that didn't crash (at least, not often). There were no standards or notions of user-friendliness, so whatever you did was as fine as anything else. Your program was finished when the programmer said it was.

That's not true now. Users *expect* human-interface standards to be followed, and the market is now big enough for you to have competitors who are trying to woo your customers away with a better product. The result: Your program is finished when users say they'll buy it. Elegant, easy-to-use programs do not spring, full-blown, from the designer's head like Athena from the brow of Zeus. They are designed and tested, redesigned and retested—as many times as it takes. ("Testing," here, means testing the human-interface design with the intended audience to see if they have any problems using it.)

This article gives you a look at the design/test cycle through one example: the evolution of HyperCard 2.0's script-editor window. Not only is it interesting from a historical point of view, but it should also comfort you to know that the *sheer living hell* that *you* go through in designing your products is also familiar to other, equally talented folks.

Motivations. Why bother changing the HyperCard 1.x.x script editor? If it works, don't fix it—right? That's not good enough for the professional, and it's not good enough for your product—the ones that stand still get left behind in today's market.

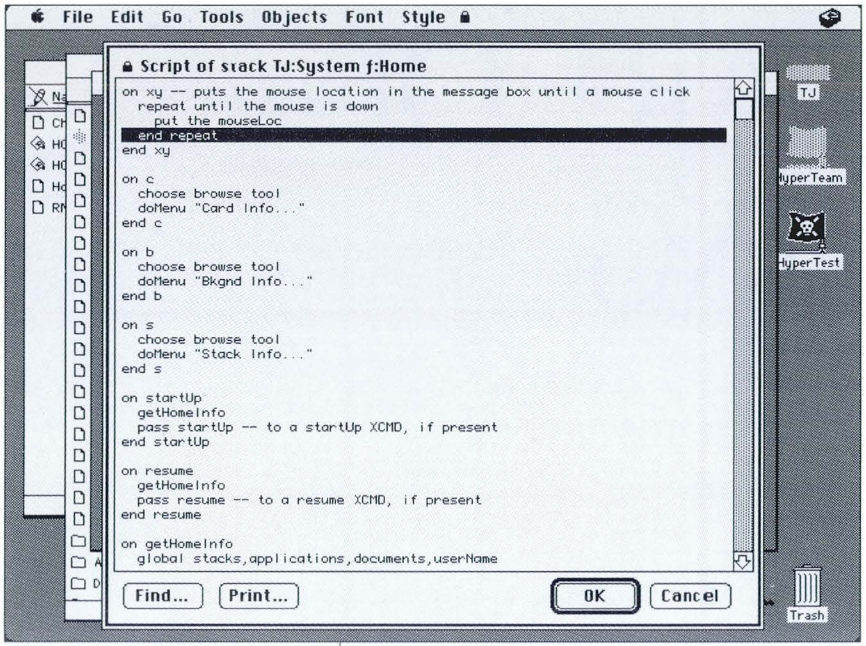
The HyperCard 2.0 team had several reasons for wanting to improve the editor. First, everybody *really* hated the old modal script editor. You couldn't move it, you couldn't see more than one script at a time, and you had to close the modal editor to do anything else. These are more than enough reasons to improve the editor, but the team wanted to make other requested improvements too.

Just when the team was thinking about the script editor, a serendipitous opportunity presented itself. They had just finished working on the extensions to the XCMD interface, which (at the time) allowed programmers to create an external window in the floating layer of HyperCard 2.0. (For more details, see "HyperCard 2.0: a Good Tool Gets Better," the cover story in last month's *Apple Direct*.)

Why not—they asked each other—implement the script editor as an external window? It was a good idea, for two reasons. First, it would give them a chance to test and improve the XCMD interface's capabilities—or, as one engineer put it, to "test it with something you care about." Second, designing the script editor as an XCMD created the possibility of a third-party market for script editors—so developers as well as users benefit.

With these goals in mind, the sections that follow give the chronology of what happened (with some conclusions at the end).

Your program is finished when the users say they'll buy it.



```
on xy -- puts the mouse location in the message box until a mouse click
  repeat until the mouse is down
    put the mouseLoc
  end repeat
end xy

on c
  choose browse tool
  doMenu "Card Info..."
end c

on b
  choose browse tool
  doMenu "Bkgnd Info..."
end b

on s
  choose browse tool
  doMenu "Stack Info..."
end s

on startUp
  getHomeInfo
  pass startUp -- to a startUp XCMD, if present
end startUp

on resume
  getHomeInfo
  pass resume -- to a resume XCMD, if present
end resume

on getHomeInfo
  global stacks,applications,documents,userName
```

FIGURE 1.
The original modal XCMD editor. Although implemented differently, it looks identical to the one in HyperCard 1.x.x.

August 1989: The HyperCard 2.0 team decided to implement two script editors—a modal editor, exactly like the one in HyperCard 1.x.x, and a new modeless editor—both using the new XCMDs and their external windows. They decided that HyperCard 2.0 would come with the modal editor, the one that behaves like the editor in HyperCard 1.x.x, installed (see figure 1). Users could install the other script editor (or a third-party one), however, by executing a HyperTalk statement such as

```
    set the scripteditor to
modelesseditor
```

The modeless editor (see figure 2) had a zoom box and the same four buttons—Find, Print, OK, and Cancel—as in the original editor, and in the same place (at the bottom of the window). The modeless editor could scroll text vertically but not horizontally. It lived in the floating layer of the screen, so it "floated" above everything else on the screen and was always "hot" (always responded to mouse clicks).

Advantages: • Current HyperCard users would not be confused by a new kind of script editor, but they could easily switch to a more powerful one if they wanted to.

• The design is "open"—that is, it encourages alternative script editors, opening up a new market to third-party developers and potentially giving users a wider variety of script editors from which to choose.

• By using the new XCMD/external-window combination, they got an opportunity to "exercise" this feature and improve it based on their real-world experience with it.

• As shown in figure 2, you could have multiple modeless-editor windows open at the same time, and you could easily resize them and move them around.

Disadvantages: • Creating both script editors as external windows meant implementing, debugging, and maintaining two editors from the ground up, with the new implementation of the modal editor requiring a lot of time and energy and having no advantage over the original implementation.

• One of the major complaints about the original script editor is that it is modal (that is, you can't work in any other window while this one is open). Most HyperCard users found the modality very frustrating.

September 1989: The team experimented with a third type of script editor, called TextWindow (based on an existing HyperCard 1.x.x XCMD window). TextWindow had all the features of the modeless editor, and it supported Undo. TextWindow became the editor of choice within the HyperCard 2.0 team—everyone liked it, and its code was already tested—so they decided to use it as HyperCard 2.0's script editor.

The TextWindow editor (which eventually assumed the official name ScriptEditor) was the first to have its own menu (as shown in figure 3—the menu items hide the OK button and part of the Cancel buttons). Find and Print (formerly buttons in the editor window) are now in the Script menu; the OK and Cancel buttons remain, but they have moved to the top right corner of the window. This editor displayed its title in a newly enlarged title bar.

Advantages: • The Script menu allowed the designers to add new features to the script editor without cluttering its window.

• With the modeless editor of figure 2, it was easy to move the window so that the buttons at the bottom were off screen. This editor solved that problem by putting the buttons at the top of the screen—if you could see the window, you could probably see its buttons.

• Putting the title of the window into the window bar and having the close and zoom boxes there too make the window look more like a modeless window than a modal dialog box, so it feels more familiar to most users.

Disadvantages: • The OK and Cancel buttons drew themselves flush right, which left an unaesthetic blank area to the left.

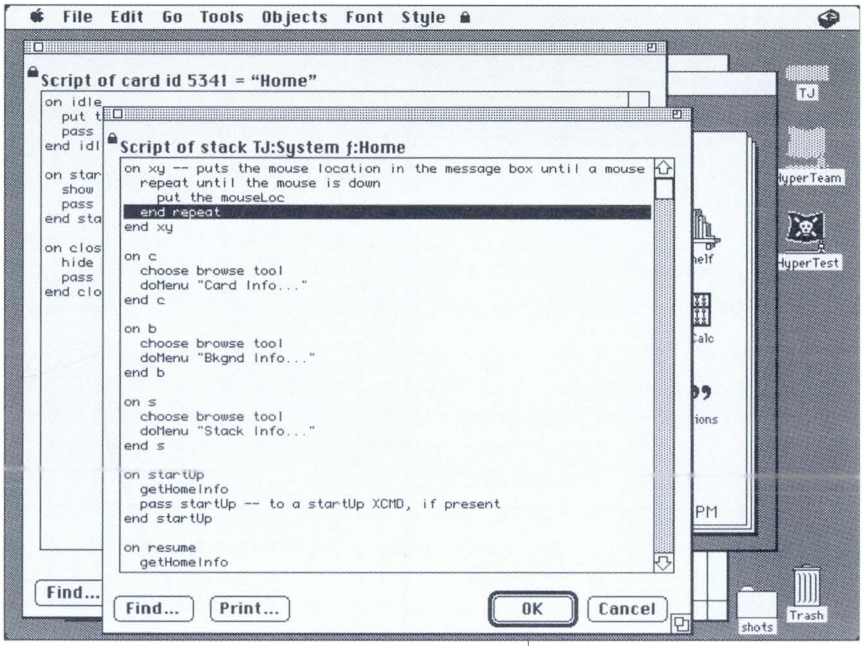
• The fact that the script-editor window was in the floating layer led to an unacceptable inconsistency in the Macintosh user interface—it acted like both a floating window and a document window.

This result came from several conflicting elements. Because the script editor was a modeless window, you could click on another window (which would redraw itself into its active state), and the script editor would draw itself as inactive. On the other hand, since it was in the floating layer, the script editor always stayed on top, even though it was inactive and on top of the active window. This runs contrary to the universally shared convention that the active window is always on top of all the other windows. What's worse, when the script-editor window became inactive, the Script menu disappeared from the menu bar, further confusing users.

Happy New Year! (early January 1990): In the last months of 1989, several things changed (see figure 4). The team "folded" into the current editor a debugger that had existed in the first modeless editor window (which had been discarded). During debugging, the Script menu is replaced by a Debugger menu. The team removed the OK and Cancel buttons and put corresponding menu items onto the Script and Debugger menus; they also added the Enter key as a keyboard equivalent for OK and command-period for Cancel. One of three symbols (a bug, a pencil, or a pencil and slash) in the lower left corner of the window told the user that it was in one of three modes: debugging, editing, or viewing only.

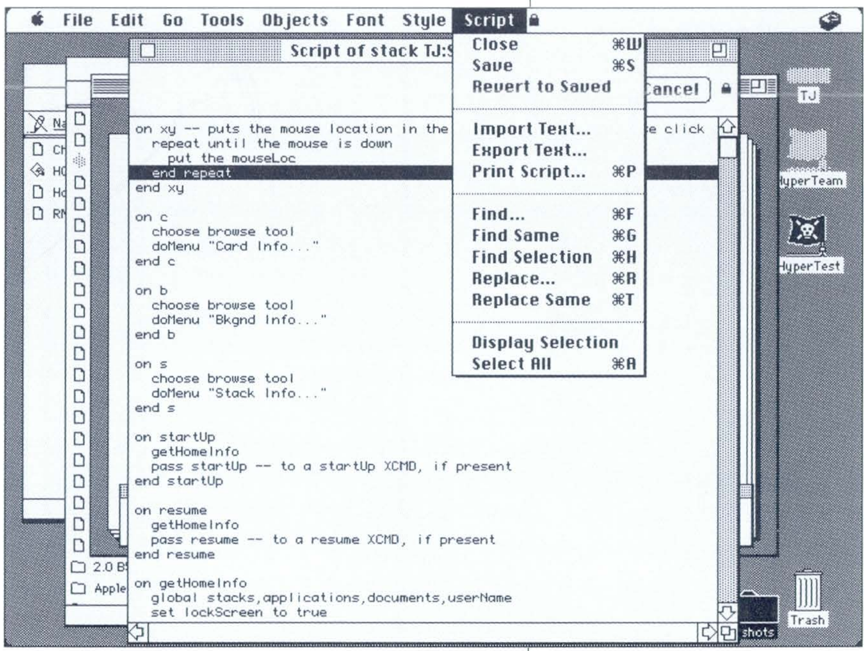
Advantages: • By using only one kind of window, the design team made things better for users (only one kind of window to learn) and themselves (only one kind of window to design, implement, debug, and maintain). The menu title and the graphic element in the lower left corner gave users two visual cues about the editor's current state.

• By moving the two buttons into the menu, they simplified the window's appearance.



```
on xy -- puts the mouse location in the message box until a mouse
  repeat until the mouse is down
    put the mouseLoc
  end repeat
end xy

on c
  choose browse tool
  doMenu "Card Info..."
end c

on b
  choose browse tool
  doMenu "Bkgnd Info..."
end b

on s
  choose browse tool
  doMenu "Stack Info..."
end s

on startUp
  getHomeInfo
  pass startUp -- to a startUp XCMD, if present
end startUp

on resume
  getHomeInfo
```

FIGURE 2, TOP
The first modeless editor. Note the addition of the drag bar and close, zoom, and resize boxes. You can also have multiple windows open and move them around.



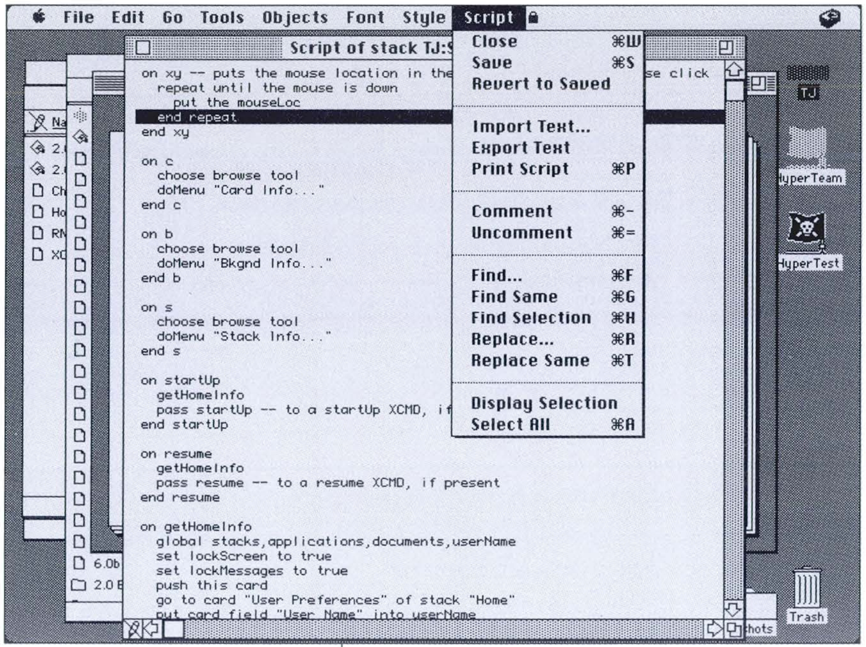
```
on xy -- puts the mouse location in the                     e click
  repeat until the mouse is down
    put the mouseLoc
  end repeat
end xy

on c
  choose browse tool
  doMenu "Card Info..."
end c

on b
  choose browse tool
  doMenu "Bkgnd Info..."
end b

on s
  choose browse tool
  doMenu "Stack Info..."
end s

on startUp
  getHomeInfo
  pass startUp -- to a startUp XCMD, if present
end startUp

on resume
  getHomeInfo
  pass resume -- to a resume XCMD, if present
end resume

on getHomeInfo
  global stacks,applications,documents,userName
  set lockScreen to true
```

FIGURE 3, BELOW
The TextWindow editor. This window looks more like a document because of the script name in the drag bar and the text field that fills the window on three sides.

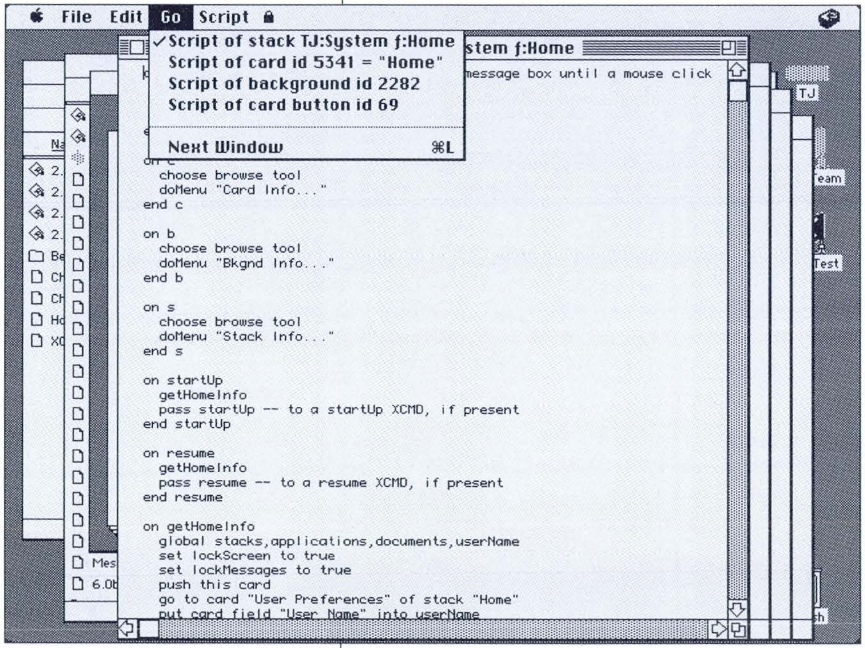
Disadvantages: • Leaving the editor window without any visible buttons might confuse HyperCard 1.x.x users, who expect to accept or reject changes just by clicking the OK or Cancel button.

• The Enter keyboard equivalent for the OK button is nonstandard and not intuitive; the graphic element is too small to be noticed and may also be confusing.

February 1990: There is now only one type of window, the ScriptEditor window, and it is used for both script editing and debugging. This is HyperCard 2.0's official editor/debugger window. The team modified the new XCMD design



```
on xy -- puts the mouse location in the                    se click
  repeat until the mouse is down
    put the mouseLoc
  end repeat
end xy

on c
  choose browse tool
  doMenu "Card Info..."
end c

on b
  choose browse tool
  doMenu "Bkgnd Info..."
end b

on s
  choose browse tool
  doMenu "Stack Info..."
end s

on startUp
  getHomeInfo
  pass startUp -- to a startUp XCMD, if
end startUp

on resume
  getHomeInfo
  pass resume -- to a resume XCMD, if present
end resume

on getHomeInfo
  global stacks,applications,documents,userName
  set lockScreen to true
  set lockMessages to true
  push this card
  go to card "User Preferences" of stack "Home"
  put card field "User Name" into userName
```


File
Edit
Go
Script
✓Script of stack TJ:System ƒ:Home
Script of card id 5341 = "Home"
Script of background id 2282
Script of card button id 69
Next Window ⌘L
stem ƒ:Home
TJ
eam
Test

```
                                  message box until a mouse click

  choose browse tool
  doMenu "Card Info..."
end c

on b
  choose browse tool
  doMenu "Bkgnd Info..."
end b

on s
  choose browse tool
  doMenu "Stack Info..."
end s

on startUp
  getHomeInfo
  pass startUp -- to a startUp XCMD, if present
end startUp

on resume
  getHomeInfo
  pass resume -- to a resume XCMD, if present
end resume

on getHomeInfo
  global stacks,applications,documents,userName
  set lockScreen to true
  set lockMessages to true
  push this card
  go to card "User Preferences" of stack "Home"
  put card field "User Name" into userName
```

FIGURE 4, TOP

An intermediate ScriptEditor window. This editor puts the remaining buttons into its menu and adds to the lower left corner a small drawing that indicates the window's mode.

FIGURE 5, BELOW

The final HyperCard 2.0 script-editor window. This window looks exactly like a document window, but while it is active, it takes over the menu bar.

so that external windows can now reside in the card layer (now called the *document layer*). The ScriptEditor window now looks and acts like a normal document window (see figure 5). Some menus disappear, and other menus' items adapt to the script window's context (the Close Stack item, for example, becomes Close Script).

Advantages: • This edit/debug window is more versatile, and it causes none of the confusion that the window in the palette layer did.

• The fact that it looks and behaves like a normal window strengthens the Macintosh human interface in the user's mind and makes using the editor easier for both new and experienced HyperCard users.

Disadvantages: Some existing HyperCard users might be confused by the new editing window, (but its appearance as a document window will give most people the cue they need to know how to use it.)

• The menu bar changes, depending on its use; this can be confusing to some users, but the behavior is consistent with the menu-bar changes that occur when they switch applications within MultiFinder.

Lessons Learned. During the process of evolving the HyperCard 2.0 script editor, the HyperCard 2.0 team solved several problems. The first was one of the human interface: What model should the editor window follow? Although it had elements from documents, (floating) windoids, desk accessories, and stand-alone applications, it was like no one type. Macintosh precedents alone could not solve their problem, and it's important to note that the solution the designers settled on has elements of each type in it and that the solution is still faithful to the spirit of the Macintosh interface.

The team also learned a lot about the new XCMD interface. They learned that originally it wasn't flexible enough, so they enhanced it to allow for external windows that live in the document layer and have their own menus.

Finally, in the evolution of the editor, the team did a good job of balancing the trade-offs in its design. They went from HyperCard 1.x.x's modal script editor to the final editor without confusing users or scaring them off. They added needed and requested features to HyperCard 2.0 without losing its simplicity. And they provided a good editor while still leaving the market open for more-sophisticated editors from third-party developers.

The fine tuning of HyperCard 2.0's script-editor window may not seem spectacular, but the changes made are very real and they will matter a great deal to anyone who ever browses or uses HyperTalk scripts. The term *fine-tuning* does not refer to small details that make a little difference—rather, it refers to ones that make a big difference. The art of fine-tuning is at the heart of successful software creation. These are the details that distinguish the personal—and commercial—success of the professional.

MACINTOSH USER TRENDS

IntelliQuest surveys 1,900 Macintosh users

By Lisa Raleigh
Apple Direct staff

Every developer wants to know who's using Macintosh computers and what third-party products those users want. Fortunately, several market-research companies are trying to divine this information, but for some developers, the price tag for professional market reports may be a little steep. The purpose of the Market Insight column, then, is to share with you pertinent market information that comes our way.

IntelliQuest, an Austin-based market-research firm, recently completed MediaTrack IQ, a study that snagged our attention. IntelliQuest surveyed 9,654 readers of computer publications—1,900 of those surveyed reported that they use a Macintosh computer. The researchers took a look at who these users are, what their purchasing intentions are, and where they obtain third-party products.

As you dive into this data, though, keep a couple of things in mind: The respondents to this survey are readers of computer publications, indicating a certain level of sophistication that is echoed by the fact that more than 60 percent of the Macintosh-using respondents report that they are advanced computer users or computer professionals. So consider the results that follow as a view of power users, or "influencers," says Rita Stewart, manager of MediaTrack IQ. In addition, because the information is not based on a survey of "all computer users," says Stewart, it is probably best to view the results as indicators of trends rather than as definitive market projections.

Consider the MediaTrack IQ results as a view of power users.

Who's Buying Mac? Figures 1 and 2 give a work-environment profile of the Macintosh-user respondents to this survcy. Figure 1 shows the size of the companies the respondents work for, as compared with the distribution of the general population in the same company-size categories (the comparative figures coming from the Bureau of Labor Statistics). You can see that these Macintosh users are heavily represented in businesses with more than 1,000 employees.

Figure 2 shows the top seven types of businesses for which these users work. Education took the most prominent position, and it is followed by six other industries that accounted for 45 percent of the responses. These Macintosh users, then, appear to represent a fairly broad range of businesses.

Along the same lines, figure 3 gives you the top six job functions of the Macintosh-user respondents—and, again, the results are pretty eclectic. In fact, the results are spread fairly evenly over the six top-named job titles, indicating again that the Macintosh is reaching a diverse audience when it comes to the magazine-reading segment of Macintosh users.

What They Want to Buy. The survey asked which applications the respondents' *organizations* intended to buy in the next 12 months (not just the individuals themselves), and it's evident from the top-ten lists in figures 4 and 5 that there continues to be plenty of interest in acquiring additional applications and peripherals of many types.

Where They're Buying. The Macintosh-user respondents to this survey obtain products from many sources, as you can see in figure 6. The chart underscores the importance of mail order

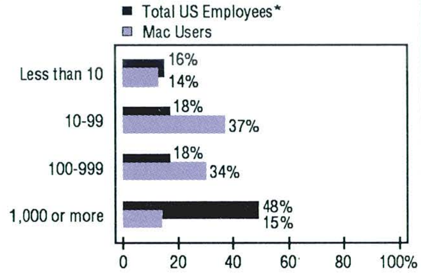
These respondents can be very influential in helping you reach other Macintosh users.

as a distribution channel for this type of user. There's also a fair amount of "direct from manufacturer" distribution going on.

The information provided here is just a small portion of the information gathered by the MediaTrack survey. Overall, the MediaTrack data points up a very broad audience for Macintosh products and a strong interest in acquiring more and more products. It's the audience you reach through computer publications and can be very influential in helping you indirectly reach other Macintosh users.

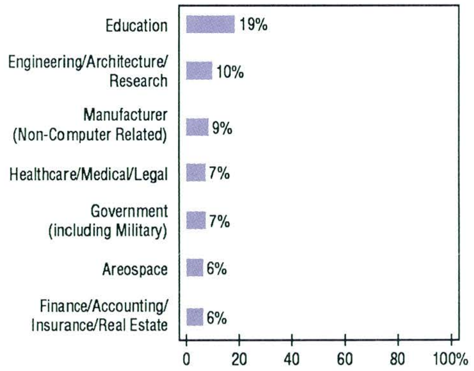
If you'd like more information about the IntelliQuest MediaTrack IQ study, write to or call Rita Stewart, Manager, MediaTrack IQ, IntelliQuest, Inc., 400 West 15th St., #815, Austin, TX 78701; (512) 320-8585.

1. Number Employed by Respondent's Organization



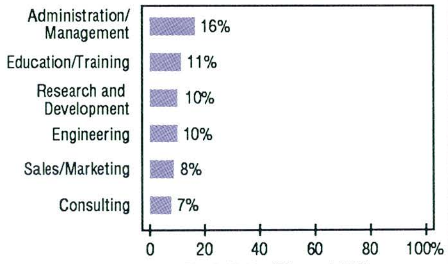
* Source: Bureau of Labor Statistics
MediaTrack IQ (IntelliQuest 1/90)

2. Primary Business Activity at User's location



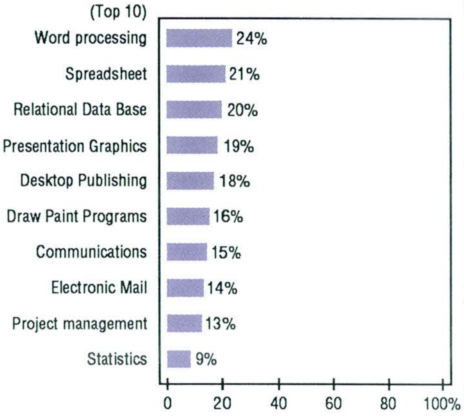
MediaTrack IQ (IntelliQuest 1//90.)

3. User's Primary Job Functions



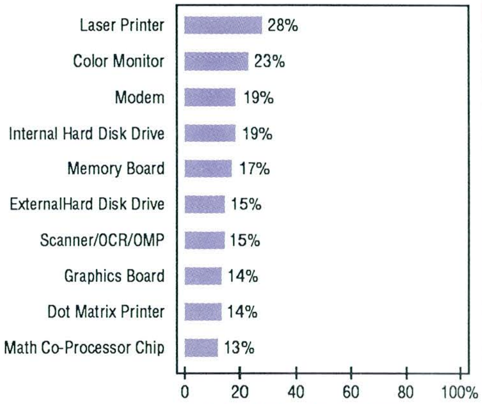
MediaTrack IQ (IntelliQuest 1//90.)

4. Software Applications User's Organizations Plan to Purchase



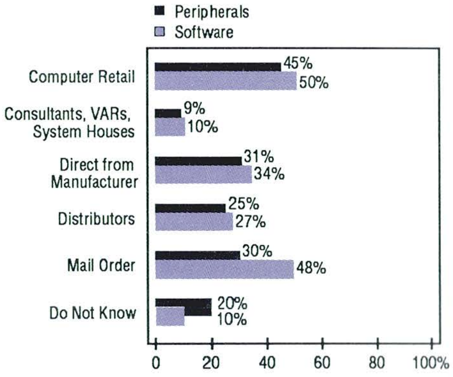
MediaTrack IQ (IntelliQuest 1//90.)

5. Peripherals User's Organizations Plan to Purchase



MediaTrack IQ (IntelliQuest 1//90.)

6. Sources for User's Organizations Software & Peripheral Products



MediaTrack IQ (IntelliQuest 1//90.)

By Tim Morgan

Voice Input and System 7.0

Imagine being able to tell your Macintosh what to do—I mean literally to drive it with spoken commands. To print a document, for example, you would simply say "print" and then perhaps the name of a file. To send a message to someone over a network, you would just say "message to John Doe" and then record your message. You would append spoken comments to documents in exactly the same way—"annotate" you would say to request the function, and then you would record your comments.

Over the past four years, Articulate Systems, Inc., has developed a system that enables you to do just that, issue spoken commands to your Macintosh. We call it The Voice Navigator. Currently it's a combination of a SCSI hardware device and software drivers. The Navigator's hardware captures and preprocesses analog voice signals, and associated software recognizes spoken commands and causes a sequence of keyboard and/or mouse events to do the actual work.

Our idea, though, is not to replace the keyboard and mouse completely, but rather to complement them with voice input and thereby make the human interface as transparent as it can be. Keyboard, mouse, and voice all have their own strengths and, naturally, their own weaknesses; what we want to do is harmonize these three input devices. Voice, as we see it, is simply a more fluid input device for some tasks and has the potential therefore to increase productivity.

Giving Voice a Language. But for voice to be an effective tool for communication with a computer, it must have access to the current set of legal commands within an application—that is, the application's *language*. Every application has such a language. It is nothing more than the means by which users control the application—menus, buttons, tool palettes, and the like. This language is quite well defined—you can think of it as a combination of sets and subsets of commands: the set of menu commands (Apple, File, Edit) and the subset of File-menu commands (New, Open, Close); the set of dialog commands (Open, Save, Print) and the subset of Open-dialog commands (Drive, Eject, Open, Cancel).

We use a desk accessory called Language Maker to create a language file for particular applications. This DA automatically reads the application's menu list and creates the language to drive all the menus. It also allows users to assign commands to particular actions or sequences of actions, such as selecting the rectangle tool from a palette. In the case of menus and buttons, we can name the commands without user intervention, because menus and buttons have text strings associated with them whereas tools in palettes don't. Today defining the language involves some setup time, but with System 7.0, applications will be able to provide this kind of information dynamically and automatically through IAC (interapplication communication).

System 7.0's *AppleEvents* are central here. By using AppleEvents, an application will be able to announce its current set of legal commands and, in return, a future version of the Voice Navigator will be able to let the application know when the user issues one of these commands by voice. For example, if the application puts up an Open dialog box, either the application or the dialog manager will tell us that only the commands Drive, Eject, Open, and Cancel are valid. The Navigator, in turn, will listen for the user to issue one of these commands and then let the application know which command was spoken.

A Call for Standards. Much of what we do today by using voice control depends on standard data structures, such as those within menus and controls. These data structures contain important information, such as the text names of objects as well as their location on screen. Crudely speaking, our voice system now resides between the application and the user; it compares what the user says with the names of these objects. Then, when the spoken word and the name of object match, it generates the appropriate events, faking the application into believing that the user has performed a mouse or keyboard action. This scheme works, for the most part, but as you can imagine, it is not foolproof and by no means exploits the full potential of a tightly integrated voice interface.

One of our most serious problems, however, is that some applications don't conform to standard data structures, and those that don't conform can't take advantage of voice in certain situations. A prime example of this nonconformity is Apple's very own Direc-

Continued on next page

Tim Morgan is the Vice President of Research and Development at Articulate Systems, Inc.

Developer Outlook

Continued from previous page

tory Title PopUp in SFGetFile and SFPutFile dialog boxes (those you get when you save and load files). The code implementing the pop-up menu doesn't use the standard PopUpMenuSelect call, so there's no easy way to get at the data structures it uses. Nonstandard menus provide another classic case of nonconformity—the Finder's Color menu, according to its menu data structure, has only one item, named x, rather than the eight color-bar menu items the user sees. It would be more helpful if it contained the number of items in the menu and, at the very least, space to store the names of menu items to assure compatibility with other software.

Another good example is a nonstandard control without a control definition function. Such a control is not really a control at all, since it doesn't use the Control Manager. Rather, it's drawn directly to the screen and only the application knows about it; there's no control record (standard data structure) for us to examine and extract information from, such as the control's name, location and state. Take a look at the speaker-volume control in the Control Panel—it's nonstandard, and only the Control Panel knows about it. When we circumvent these standards, we lose valuable information and that loss hampers the creation of a more transparent interface.

One solution to this problem, of course, is for us developers to avoid program dependencies—provide as much information as possible in standard resources, especially when there is no reason not to do so. And don't assume anything. If tool palettes—not just their look and feel but their associated resources too—were standardized, we'd know which was the rectangle tool even if the user didn't point it out.

The Integrated Voice. Lately, though, some of this reluctance to conform to standards has begun to change, and we are currently working with several developers to integrate voice into their applications. The real barrier to this kind of cooperation is the lack of a common protocol across applications. And this is where System 7.0 again proves to be essential.

I have always believed that for voice to become a standard means of communication between humans and computers, it would be necessary to build it into the operating system. With System 7.0, Apple has provided the building blocks to make this a reality. Developers can now establish standard protocols for communicating with one another via AppleEvents, and once established, these events can be shared by all cooperative applications.

The key issue is context, an issue developers themselves must decide with respect to their applications. For example, the spoken command "Open" is, in and of itself, ambiguous. There is generally not enough information in the isolated word to determine with certainty just what it means. In this case, there can be three different interpretations: The user has preselected certain objects and now wants the applications to open them, there is a ModalDialog on screen with an Open button, or neither of the above.

The application's response to the spoken command will be slightly different for each condition. If the application is confronted with the first or second condition, it may simply open the selected objects. If the application confronts the third condition, it might wait a certain amount of time to allow the user to name the object to be opened; then if nothing happens, it might display the standard Open dialog to prompt the user for more information.

Only you are privy to the kind of commands needed to drive your application at any particular time. When a user saves a document, you must determine whether or not a SFSaveDialog is needed (if the filename is untitled). Certainly the Navigator, or a similar system, could trap every dialog call to keep track of the application's state, but that would be uneconomical and also a waste of the opportunity System 7.0 provides.

The Conversational Desktop. The following is an example of what might happen with the combination of technologies I have been discussing in conjunction with IAC.

User: "Open *Apple Direct* article."

The Finder launches MacWrite and tells it to open the Apple Direct *article.*

User: "Select paragraph 3."

MacWrite highlights the third paragraph.

User: "Annotate."

MacWrite displays a recording gauge while the user makes her comments. When the user has stopped talking, MacWrite displays a dialog box and asks,

Mac: "Have you finished?"

User: "Yes, send."

A couple of moments pass while MacWrite checks to see if there's an application that can carry out the request. Microsoft Mail comes to the foreground, displays an address book, and asks,

Mac: "Who would you like to send it to?"

User: "Larry Tesler"

Microsoft Mail knows the address, the file is sent, and the transaction is complete.

Now this may seem a little far-fetched, but there is nothing in this scenario that couldn't take place in a System 7.0 environment. In the current Macintosh environment, it's possible for one company to provide a voice solution that all third parties can use. In fact, for programs that "obey" the Macintosh interface rules, providing voice can be quite straightforward. With System 7.0, however, it will be possible to incorporate voice in the interface in a completely transparent way. For this reason alone, you should install System 7.0's custom AppleEvents in your programs so that other programs can use them to get their work done. Voice I/O will happen on the Macintosh; with your cooperation, it will happen quickly and easily.

By Carolyn André

How to Create Effective Advertising

Advertisers spend billions of dollars each year pushing products. Yet many of them, possessors of large and small ad budgets alike, complain that the ads for which they pay so dearly don't do enough. And, alas, they're frequently right.

True, advertising that sells is often divinely inspired, but effective advertising requires more than divine inspiration. For an ad to be effective, it must sell the right features of a product to the right audience, in addition to being clear and well crafted.

That sounds simple enough. So why, you're wondering, does so much advertising fail to do the job it sets out to do? Naturally, there are lots of reasons—too many reasons, in fact, for us to cover all of them here. But in my experience, most advertising fails because clients don't give their advertising agency the right input *before* the agency's creative team puts pen to paper.

At the very least, that input should include the following three components: well-defined goals for your ad, a strategy based on where your company is now and where you want to take it, and a true understanding of the target market and how that market perceives your product.

Setting Goals. Let's start with goals for advertising. Naturally, your advertising goals will vary, depending on your strategy, how long your company's been around, your product history, and the way you're perceived in the marketplace vis-à-vis your competitors. If you're a start-up company, for example, your initial goal is typically to make people aware that you're in the marketplace. If you're a well-known company with proven products, you may want to persuade the target audience that your new product is better than the competition's. If the fiscal year is closing and you're falling short of your sales goals, you may want a tactical ad designed to push boxes out the door.

You would assume that deciding just what you want your ad to achieve would be as straightforward as death and taxes, but that's not necessarily so. You'd be amazed by the number (not to mention the size and prestige) of advertisers whose advertising goals are vague.

Some time ago, I did some work for a well-known computer magazine whose publisher told me it was losing a major advertiser. The magazine had run one of the advertiser's full-page ads for several weeks, but the ad had not delivered the number of phone calls the advertiser had anticipated. Naturally, the advertiser blamed the magazine. The fault, however, lay in the ad itself.

Knowing what you want makes deciding whether you're getting it a lot easier.

That ad defied you to find the toll-free number buried within it. When I asked the ad's creator why he'd hidden this crucial component, he told me that he hadn't been asked to create a direct-response ad. "It was an image ad," he said, "and at the last minute the client wanted me to add a toll-free number."

If you don't spell out your goals (in this case, the undisclosed goal was to keep that direct-response phone ringing), scenarios such as this one will continue to be the rule rather than the exception. An image ad and a direct-response ad are very different animals—they look different from each other, and they do different things. It's up to you to determine what you want the ad to accomplish, and it's your responsibility to convey that to your ad agency.

Deciding Strategy. Once you have your goal in mind, you need a clearly articulated vision of how you want your company and products to be perceived. (Remember: Knowing what you want makes deciding whether you're getting it a lot easier.)

I recently evaluated two alternative ads for the same product. One ad created the impression of a state-of-the-art product, hot technology from a modern, go-get-'em kind of company. The other gave the impression of a reliable, low-risk product from a company that was clearly on the outskirts of technology's forefront. There was nothing inherently wrong with either image. The company was known as large, stable, and reliable, but it suffered from rumors that its technology was no longer up-to-date. The ad that created the impression of a state-of-the-art, hot product, therefore,

Continued on next page

Carolyn André is president of André Associates, a strategic-planning and research firm in Berkeley, California.

Marketing Clicks

Continued from previous page

was perhaps the more appropriate of the two. However, the client and the agency couldn't agree on which ad to run, because neither had articulated what the ad was intended to do.

Understanding Your Target. Once company goals and strategy are set, it's critical to position your product so that your target market can relate to it. To do that, of course, you need to understand that market.

Understanding, I mean *really* understanding, your target market is more difficult than it sounds. True understanding goes beyond the numbers so readily available from today's demographers and statisticians. Numbers don't always speak clearly, but the members of your target market will. So go to them. Ask them how they perceive the market, how they think your product fits in. Ask them how and why they buy what they buy, and last, ask them whether they understand the message you're trying to convey with your ad.

Now, I don't mean to suggest that you should stick a sketch of your ad in some unsuspecting soul's face and ask, "Is the main idea of this ad understandable to you?" No. That approach will lead directly to a dead-end yes-or-no answer—it won't get you anywhere. Instead, you might present your ad to members of your target market and ask them to identify its main idea. If you get several responses you didn't anticipate, it's back to the drawing board.

And while you're probing your market, be sure to listen to what your respondents *mean* rather than just what they *say*. For instance, you're selling a hardware product for which connectivity is probably an issue. You approach members of your target market and, after showing them your ad, ask, "If you were to buy this product, would connectivity be an issue for you?" Your respondents might say, "No, connectivity is not a problem." But what they might well mean is "Connectivity is not a problem, because if the product doesn't have the connectivity I require, I won't even look at it." Listening for the message underlying what your target market tells you is essential.

Knowing Your Competition. To create an effective ad, you must also understand how your product is perceived in comparison with your competition's product. And you need to interpret the word *competition* in the broadest sense. A product always has competition. Even the first personal computers had competition, and I'm not referring to the VAX or mainframes from Big Blue but rather pen and paper. Ask yourself what the members of your target audience are doing now without your product. Ask yourself how they are doing it, and then find out which products they're using to do it. Conveying a knowledge of your competition in your ad never hurts.

This becomes even more important as product categories blend. For example, you've developed a product that combines a relational database, a spreadsheet program, and a word processor with drawing capabilities (wisely, you've left out the kitchen sink). How do you position such a product? Your competition may not be immediately apparent to you. However, a conversation with your potential buyers will help identify products *they* perceive to be competing with yours. By discovering how your potential buyers view your product, you might also more thoroughly understand what its major benefits are.

Attending to Detail. For an ad to be effective, it has to be relevant and understandable to the target market. A simple example: You're a software manufacturer selling direct. Does your ad list your products by category or publisher? Well, that depends on what is easiest for your target audience to recognize. If you're going to list your products by publisher, you'd better be sure the names of all the publishers are well known.

Also bear in mind that the design of your ad should clearly represent your product. If, for example, you're known for a DOS software product that you are now introducing for the Mac, you might consider changing the look of your packaging and your advertising. Your bronze-and-black color scheme may be easily identifiable, and it may therefore seem logical to extend that look to your Mac product. But beware. Purchasers often tell me that they expect a Mac product to represent the machine it runs on. That implies color, a graphic look, and a message that's quite different from what you'd deliver to DOS buyers.

Adding It All Up. In general, people don't want to work to understand your ad. Your message must be immediately recognizable and relevant to your market, to their problems, to the benefits they seek. You can't guarantee success, but you can radically decrease the possibility of failure by defining your goals and your strategy and testing your hypotheses out on the audience you're selling to.

By doing so, you may well avoid joining the legions of advertisers dissatisfied with what they get (or don't get) for their money.

Cover—Looking Into the Future

Continued from page 12

put off thinking about the future. Products *now* are using sound—what will they do in a year? What will your products do in a year? Two years from now, superior products may use animation, maybe even video. What are the most likely directions for your products in that time frame?

The possibilities for future software are breathtaking, and I'm sure that a handful of products, maybe more, will be marvelous in ways that nobody has thought of today. You and your company have your own viewpoint, one that no one else in the industry quite duplicates. Think of these new technologies; what do *you* see? Turn that into a product, and both you and the entire industry will be richer.

By Ben Sharpton, Tupperware

Customer Service—From the Customer's Perspective

In *Five Easy Pieces*, there's a classic scene in which Jack Nicholson and some friends order breakfast in a little café. Jack orders first. He wants a couple of eggs, some hash browns, and a side order of toast. The waitress explains that they don't serve toast and don't offer side orders. She offers to return when Jack's made up his mind. Jack explains that he's made up his mind and that he wants some eggs and toast. The waitress points to a sign that says, "No Substitutions."

By now, both are flustered. In his grating fashion, Jack meticulously explains that he wants to order an egg. And a chicken-salad sandwich: "Hold the lettuce and mayonnaise." The irritated waitress repeats his order back to him. "One egg, one chicken-salad sandwich, hold the lettuce and the mayo. Is there anything else?"

"Yeah," responds the rebellious Nicholson. "Now hold the chicken salad and bring me two pieces of toast." The waitress says incredulously, "You want me to hold the chicken salad?" to which Nicholson says, "Yeah, I want you you to hold it between your knees." With a sweeping motion, he knocks all the plates, glasses, and silverware off the table, and he and his friends storm out.

I think I know how Nicholson felt. I've seen poor and inflexible customer service before. Unfortunately, I've experienced it with some Macintosh product developers.

Hard-Drive Errors. A month ago, one of the external hard drives in our network began to act funky. Writing errors abounded. Like most Macintosh users, I tried running utility programs in traditional do-it-yourself fashion to repair the problem.

I dialed up one of the computer networks and downloaded some programs designed to verify SCSI ports and rewrite drivers. The information about one program indicated that it was uploaded from a hard-drive manufacturer with its permission. Although my drive was from a different company than the one that had created the program, I thought that utility might help me solve my problem. I was desperate, and the documentation that came with it indicated which options might cause me to lose data on my disks.

Careful to avoid those dangerous options, I ran the program on a good hard drive. Nothing happened. I ran the program on the questionable drive. Nothing. I rebooted both drives and… Nothing. No disk-drive icon. No data. Nothing.

I called the (non-toll-free) technical-support telephone number, was promptly placed on hold, and waited an interminable period for someone to answer my call. Finally someone offered to help. I explained my situation, and he promptly told me that the program was only for people who had purchased his company's drives. It shouldn't have been on the network in the first place, and it shouldn't have been used with my type of drive.

But it *was* on the network, I explained, and it had eaten two of my drives. Could he please help me?

I buy more than your product. Like links in a chain, your product is no better than your weakest area of service.

Paying Enough. The technical "support" person explained that there *was* a way to resolve my problem but that he couldn't help me since I wasn't a *paying customer*. I figured I'd paid enough, having lost more than 100 megs of data, but he adamantly refused to help.

You know, the irony of this situation is that I never will be a paying customer for that company. In one brief 15-minute phone call (actually only about 3 minutes of conversation), this individual lost my business for a lifetime. That's unfortunate, because I'm always on the lookout for new products. What's also unfortunate is that I've received similar treatment even when I've been a "bona fide" customer. Everyone loses.

Customer Service Sells. Customer service is a selling point. If I like the way your product serves me, I'll buy it and recommend it to others. The developer that provides good customer service is a

Continued on next page

Ben Sharpton is a training writer at Tupperware Home Parties in Orlando, Florida. He is also responsible for providing technical support and training for the personnel in the sales-development department, where the standard computer is Macintosh.

Dear Developer

Continued from previous page

step ahead of the competition.

We learned about customer service years ago at Tupperware Home Parties, where I am a training writer. We chose direct marketing (you can buy Tupperware-brand products only from independent dealers) because it provided a way to demonstrate how to get the most out of our products. Our customers don't just purchase plastic. They also buy customer service. And we give them the best of both.

Customer service is far more than the traditional concept of technical support. It permeates every aspect of your business. In fact, serving the customer *is* your business. It's found in promotional materials, user manuals, and on-line help. It's a part of honoring your announced release dates. It's in the way users work with your product and the way your product works with other products. It's in your warranty. When I buy your product, I'm buying all of these. Like links in a chain, your product is no better than your weakest area of service.

Customer service is also a reselling point. If you provide good customer service, I'll look to your company to provide future products or upgrades I need.

Since the early 1950s, Tupperware has offered a lifetime warranty on all of its products. We gladly replace any Tupperware-brand product, should it crack, peel, chip, or break, as long as you both shall live. Whenever someone contacts one of our independent dealers to request replacement of an item that is not working properly, that dealer has an opportunity to introduce and sell more items. People continue to buy new products from their Tupperware dealer because they are happy with the customer service they've received with other products.

Three Foolproof Suggestions. As you evaluate your company's customer service, may I recommend three guidelines to keep in mind:

1. *The customer is right.* Period. That also goes for your *non-paying customers,* because they may become paying customers if you offer good customer service.

2. *The price is right.* I pay for customer service when I buy a product. I like dealing with a company that doesn't expect me to pay extra to make its product work correctly. I appreciate companies that use toll-free technical-support numbers and don't think of technical support as an add-on (I'm not interested in 900 numbers or support agreements).

Along the same lines, I like reasonable upgrade policies. Paying for bug repairs is unreasonable (and bad customer service). And when I order an upgrade that fixes more than the past release's problems, I don't expect to pay 40 percent of what I initially paid just to get the latest and greatest version. Two or three upgrades at that rate, and I might as well go out and buy a competitor's product.

Continued on page 29

It Shipped

The following is a listing of the companies that participated in the "It Shipped" Program between March 29 and April 26.

Affinity Microsytems Ltd.—**Tempo+ Tools**
Air Land Systems Corp.—**AMIE II Uniscope**
Arborworks, Inc—**Time&Money**
Avatar Corporation—**MacMainFrame Coax Gateway**
Burnham Consulting Group—**PowerCurve**
DataSmith—**Nexus**
Diamante Software—**Control II**
Digital Concepts, Inc.—**4thRIGHT! Professional**
Eastgate Systems, Inc.—**Fontina**
Edmark Corporation—**Edmark LessonMaker**
Electronic Learning Systems, Inc.—**Menu Master**
Evatac Software—**Pr editor**
GE Government Systems—**Radar Display Unit**
glps Products—**Bilingual Solutions for Russian, Czech, Polish, Hungarian**
Interval Music Systems—**PROTEZOA**
Jeffrey S. Shulman—**VirusDetective**
Johnson & Johnson Design/Build—**DrawTools**
Le Choft—**MacTex II**
Manor of Micro, Inc.—**StackMate**
Maynard Electronics —**MaxStream**
Mesa Graphics Inc.—**Mesa Graphics Plotter Utility**
Micro System Options—**3d Graphic Tools XCMD, 3d Graphic Tools**
Neotech Ltd—**Neotech Image Grabber**
Now Software (formerly SmethersBarnes)—**Now Utilities, Screen locker**
P3 Software - A Division of Arminus Publications and Products, Inc.—**Right-to-Know (California and New Jersey)**
Packer Software, Inc.—**Small Business/Retailer**
Page One Graphics—**FH Full Borders Collection**
Prometheus Products, Inc.—**ProModemII, TravelModem**
Qed Technologies—**Credit Profiler!**
Salient—**DiskDoubler**
Selective Memory—**List Learner**
Stevens Creek Software—**The Athlete's Diary**
Strategic Studies Group—**Panzer Battles**
Suick Bay Technologies—**MacShell**
Symantec Corporation—**THINK Pascal**
Synergy Software (PCS Inc.)—**VersaTerm v4.1, VersaTerm-PRO v3.1**
Synex—**MacEnvelope Plus**
Textco, Inc.—**Gene Construction Kit**
Three-Sixty Pacific, Inc.—**Sands of Fire, Warlock**
Thunderware, Inc.—**LightningScan 400 with ThunderWorks**
Timeslips Corporation—**Timeslips III**
Victory Software—**2088: The Cyrllan Mission**

THE HUMAN INTERFACE

By Bruce “Tog” Tognazzini

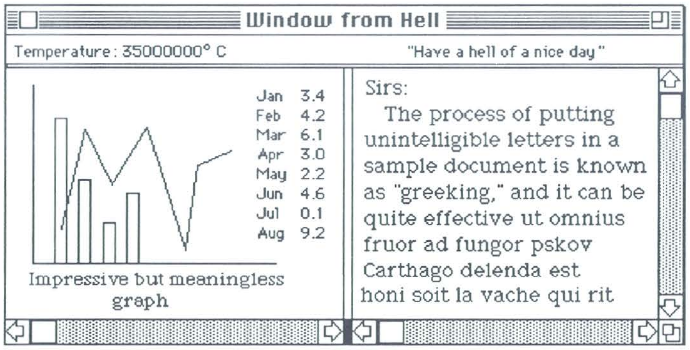
OOP Is All Greek To Me

Object-oriented programming environments are both new and difficult to create, but we should see most of their structural limitations fall away as time goes on.

Dear Tog: I feel as if I am under attack by forces clamoring for me to use object-oriented programming. Their exhortations are frighteningly similar to sermons that offer only vague promises of the rewards of the afterlife but provide exquisitely detailed descriptions of the suffering of the wicked, to wit:

“See this full text-editing program? Well, it took me only one and a half lines of code to write it. Besides, if you don’t use this new system, you’ll be totally unemployable in mere months and will find yourself lying in a gutter somewhere. The New Breed of Magnificent Young Programmers will make sport of your total incompetence as they spit in your face.”

Does this sound appealing to you? It doesn’t to me either. I think my problem is the incredible zeal with which the OOPers are pushing their product. Granted, it’s great to have the system’s default objects and methods take care of ordinary windows for you. What, however, do you do with a window like this one, with the vertical scroll bar controlling both panels:



Window from Hell

Temperature: 35000000° C

"Have a hell of a nice day"

Sirs:

The process of putting unintelligible letters in a sample document is known as "greeking," and it can be quite effective ut omnius fruor ad fungor pskov Carthago delenda est honi soit la vache qui rit

In the system I’m using, it took me more than half a day to figure out exactly where to override one of its default methods to tell it, “It’s OK to have a scroll bar that doesn’t go all the way across the window.”

I think my biggest problem was that I started believing the hype that Object-Oriented Programming Is Where It’s At and was unpleasantly surprised to find that I still had to do a lot of the dirty work myself.

I’ve been through this before. I’ve been told I was a stick-in-the-mud if I didn’t learn FORTH. I was told I’d go bald if I didn’t use PROLOG. (I am going bald but suspect that heredity and age have more to do with it than predicate logic.)

Is this new object-oriented programming a religion or a fashion statement? Or will I indeed turn green, scream, and die if I don’t jump on the bandwagon?

Awaiting your advice on the matter, I remain sincerely yours,

— *Stupefied in San Jose*

Tog Responds: I read an article some years back written by a programmer decrying the arrival of the first high-level language. He spoke in anguished tones of young Turks proclaiming the righteousness of the new approach in a fervor bordering on fanaticism. He was not against “progress”; he was against barriers that would separate him from communicating with his machine on the most intimate level. The new language, by generating a purely artificial construct through which he had to communicate, isolated him from the kind of deep understanding he had maintained with his computer. Sound familiar?

This article was written in the 1950s, and he was speaking of the transition from machine language (raw, binary numbers) to assembly language (three-character mnemonic substitutes for the

Continued on next page

This column is your input into the Apple system. Send Tog questions, comments, tirades, and editorials on the state of Apple’s human interface. Include your name, city, and state, plus company name, if you wish. He does not have the time to respond to each letter personally, but he will print those that strike a nerve and pass the others on to Apple Engineering. Rest assured, your words will not go unnoticed.

The Human Interface

Continued from previous page

same numbers). Don't laugh; that programmer had a valid point: The pattern of each number had corresponded to the structure of the processor. He had learned what effect each binary digit had on the processor, so he could build commands at will without having to memorize any of them. If he wanted to load something rather than store it, he had only to turn the second binary digit on. To do the load indirectly, he had merely to leave the third binary digit off. With the advent of letters unrelated to that structure, he had to memorize each and every command, mnemonic though they might have been, and he lost the relationship of one command to another, communicated formerly through their binary representation.

Formerly, he had known where every bit of code existed in memory. Suddenly, he had to depend on the system to allocate memory for him. He could no longer so easily prevent collisions caused by inadvertently using the same memory location for two different purposes, now that he was "protected" from knowing where things were.

It is difficult to imagine that one could create, let alone maintain, a system as complex as the Macintosh using machine language. Nonetheless, I share Stupefied in San Jose's concerns: Every step up in power has seemed to bring with it ever more restrictions on how languages could be applied. Some languages, such as Pascal, were even created for the express purpose of blocking people from programming as they wished. A supposed student language with built-in "discipline," it was forced on those of us in the real world who like to play around as we code, trying different ideas. (Playing "what if?"—what a novel idea!)

Pascal's rigid structure was there to force students to write lock-step, top-down, structured code, rather than the kind of loose, wandering spaghetti code they might otherwise have produced during their early months of programming. I could argue the merits of such an approach to teaching, but I will sidestep that discussion. (I learned quite quickly on my own not to write spaghetti code after I wrapped myself up in it until I looked like a mummy.) But I would argue that programming environments with heavily restricted structures invariably stifle innovation and creativity. I remember the struggle I had in order to "trick" Pascal into letting me create the kind of live interaction I have always employed in my programs. I was able to create state machines and live, interactive animations, but only by swimming upstream against the tide of the natural structure of the language. When one is spending most of one's time attempting to "go around" a language to get one's work done, that language, in my opinion, has failed.

Unlike Pascal, OOP environments are limited today not because of any deliberate attempt to discipline us but by the magnitude of the problem. OOP environments are both new and difficult to create, but we should see their structural limitations fall away as time goes on. Yes, developers often quickly reach the limits of OOP systems, as has Stupefied in San Jose, but not because the developers of OOP languages sat around a table and decided that people who want multiple scroll bars are bad people. OOP developers just haven't gotten to the seventh release, into which they will undoubtedly put resizable scroll bars.

Object-oriented programming has the potential to be different—eventually. One of the beauties of the Macintosh is that de-

Tog's Advice to Developers of OOP Languages:

Developers need programming languages that help them do what they do best—create. Object-oriented programming languages are the best we've seen so far, but even they can improve. Here are my recommendations for a better OOP environment.

1. Provide the host computer's standard objects. An object, in this context, consists of sensory (visual, aural, etc.) and behavioral (feedback and resulting action) elements bound together into an entity. It is not enough to provide pictures of scroll bars and some loose code that may or may not be attached to them to provide the scrolling. Provide the complete behavior.

2. Enable developers to modify objects. If a scroll bar has a tendency to fill the entire length or width of a window, the developer must be able to get inside it to change that behavior. If a developer needs a new object that is semantically related to an old one, the developer should be able to reshape the old object's appearance and behavior to create the new one.

3. Enable developers to get outside the environment. It's the old naive-user-versus-experienced-user problem: To keep new users from becoming frightened, developers lock them away from the "big system" out there. That works great, as long as newly experienced users don't discover six months into the creation of a new application that this protected environment just won't let them accomplish something and at the same time prevents them from getting to the tools that will.

4. Make it easier to do things "right" than "wrong." Interface systems should enable—even encourage—innovation but not spawn a lot of spurious variation. Most programmers want to get the job done in the most straightforward way possible. With standard objects, those programmers will automatically generate applications that do things the standard way. By enabling objects to be modified and the boundaries of the system to be breached, innovators and programmers with tough new application areas will not be stifled. Still, it will be more difficult for them than just doing it "by the book," as it should be.

velopers are free to go around any single feature of the interface, if need be, by simply not using that particular black box. I have seen nothing inherent in OOP environments that precludes this same approach; OOP designers, I trust, will eventually structure their systems to be less restrictive, perhaps by building OOP systems out of ever-smaller objects, so that we can easily take a larger object apart and put it back together in a slightly different form to fill our special needs.

Fire, Brimstone, and Object Programming. Several years ago, I visited the hydrothermal area in Rotorua, New Zealand, a country in which sheep actually outnumber lawyers. A few well-placed signs in a park filled with boiling mud pots and sulfur geysers explained that falling into a mud pot or geyser would not only prove painful but also cause friends to shun you because of the sulfur smell you would take on by so bathing. Armed with that information, visitors were then perfectly free to walk anywhere they wanted, including into the depths of a boiling mud pot. Strangely enough, the day I visited, people stayed out of the mud pots in droves.

A few months later, I visited a similar area, Bumpass Hell, in Lassen National Park in California. Due to the sheep/lawyer ratio maintained in the state of California, the Feds were taking no chances: People were being herded onto a wooden platform edged with steel fencing, located somewhat near the area where the neat stuff was going on. From a distance, I saw the mud pots and fumaroles, but I did not, in any sense of the word, experience them.

I had experienced them before, however, when I was a kid. Then America also had more sheep than lawyers, and I was free to fully experience Bumpass Hell, which I did by stepping into a foot of sulfurous mud. I wasn't burned, but my friends did shun me for some time. And I wouldn't trade that experience for anything.

Object-oriented programming has the potential to take the form of a wood-and-steel barrier or of a few signs and some sulfurproof boots (uh, I mean "sulfur-resistant"—no warranty expressed or implied). I will gladly trade the occasional high-top full of sulfurous muck for the ability to create a no-compromise application.

I think we all want an easier, more natural way to interact with our systems. We want a development environment that encourages, rather than fights, the kinds of structures we need to create within our own applications. But we don't want to be talked down to, we don't want someone else's vision of the one right way to do things woven into the fabric of the environment, and we don't want protective pathways with steel fences keeping us from getting down to where we need to be to accomplish our work.

As for the rest of Stupified's question, as to whether the dervishes seen whirling in the brilliant glow of each new language are swept up in a fashion statement or a religion, I think "cult" would be closer to the truth.

My link address remains: TOG

Dear Developer

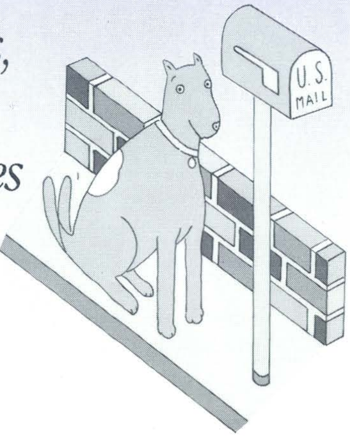
Continued from page 26

3. *Good customer service makes the product right.* Vaporware is not good customer service. Buggy software is not good customer service. If your product doesn't perform as you say it will, I'm not interested in it. If I do buy it and it doesn't work right, I probably won't be interested in your next one.

Fool Me Once, Shame On You... About three years ago, a company released a companion product to a word-processing program for the Macintosh. Two months before its product was released, the word-processing developer released a major upgrade, and the companion product was incompatible with the new format. I made the mistake of buying the companion product and waited months for it to be upgraded. The developer promised compatibility in a new, soon-to-be-released upgrade, but the upgrade never came.

Recently the same company introduced a new, unrelated product for the Macintosh. I seriously doubt that I will ever consider that program, regardless of its merits, because I received poor customer service with its maker's other product.

The way you treat me as a customer—or as a potential customer—affects the way I'll view you as a developer for years to come. Truth be told, I, the customer, am not too forgiving. I want to be treated with the respect I deserve. If I don't receive it, I'll look for other companies to offer it to me.



VIEWPOINT

By Scott Converse

The Organic Nature of Information

"The problem was how to generate a low-maintenance, high-yield, self-sustaining, critical information service."
—*S. Brand,* from The Electronic Whole Earth Catalog

Electronic media will probably be one of the most important tools Apple uses to help developers make their way through the ever-growing data jungle. We want to create an environment that helps you put data into a context useful to you—that is, turn raw data into information that in turn can be converted into knowledge.

There are many ways to accomplish this—to build knowledge or grow it. I choose the latter metaphor. Have you ever noticed how the data on your hard disk proliferates? How even untended folders fill almost of their own accord? Data tends to be organic; it grows and spreads. Even the afflictions associated with it tend to have organic metaphors—the computer virus, for example.

Apple's goal is to create an electronic garden in which knowledge can blossom out of raw data and information. And perhaps along the way, we can plant seeds that you can fertilize and use in growing your own patch.

The CD-ROM Advantage. Although electronic media include a variety of technologies (such as on-line systems and hard-disk and optical media), CD-ROM is one of the richest tiers of the electronic garden. It will be one of the most important information-dissemination vehicles during the next decade. One of the main reasons is economics: At $2 or less per disc, CD-ROM will be less expensive to create and use than almost any other medium, including paper. A plethora of CD-ROM-mastering facilities that master thousands of discs every day has sprung up. Through sheer numbers, the price of mastering a disc has been driven down, so that now you can do so for as little as $1,500. The price of drives is likewise plummeting.

CD-ROM will thrive also because of its versatility and suitability to the tasks at hand. CDs are information magnets; because so much data can be packed onto them (600+ megabytes right now, the equivalent of 250,000–300,000 single-spaced typed pages), you can do things you never before dreamed of doing. Because it's electronic, the information on a disc is malleable and can be manipulated in dozens of ways and create deep hypermedia webs with hooks into other data or information resources.

Creating Hooks. Creating hooks or links with information outside of a single CD-ROM is profoundly important, because data and information aren't ends unto themselves; they are the building blocks of knowledge. A first step in the transition from raw data to knowledge is collecting, in pivotal places, the hoards of data needed to create great products. Then the data must be put into a context, given some kind of rhyme or reason. When *raw data* is put into an easily accessible place (such as a CD-ROM) and edited to fit the needs and desires of a specific audience (developers, for instance), it becomes *information*—its new anatomy is more sensible, palatable, and useful to individuals. Crafting those pieces of information into meaningful and new insights and understanding begets *knowledge*.

Apple has taken the first step in this process—information collection—by putting significant amounts of Apple's technical documentation (*Inside Macintosh, Tech Notes*) and other important tools (system software, HyperCard stacks, Q&As from DTS, and source code) on CD-ROM, via the Developer CD Series and the *develop* technical journal CD.

This is only the start, however. Electronic media (the garden) should transcend CD-ROM. Creating hooks between these and other pieces of electronic information is what growing the garden is all about. If we create seamless links between the more static information on CD-ROM and the more dynamic information from on-line systems such as the AppleLink® network, you could begin to create a real synergy between seemingly disparate pieces of information. By putting static information (such as the technical information in *SpInside Macintosh*) onto a rather permanent medium (CD-ROM) and linking it to a constantly changing medium (an on-line system), you create something much more significant and useful. In essence, you create the basis for converting data and information into knowledge.

We plan to use the Developer CD Series to create and experi-

Scott Converse manages the Electronic Media Group, which is responsible for the Developer CD Series *and the* develop *technical journal CD. He has been at Apple for three years and was previously a manager of strategic technology development for Apple's channel systems group. He is also the coauthor of the Electronic Media column that appears every other month in* Apple Direct.

ment with prototypes for such organic hypermedia webbing. Since HyperCard is our CD search engine, we think it makes sense to expand its use to on-line systems such as AppleLink. So we're examining HyperCard/on-line-system hybrids (an example might be a HyperCard stack with all the functions needed to access the AppleLink host server).

This creates several intriguing possibilities. Such a system could transparently log the user onto a network to continue an information search and direct the search to the most obvious place (for example, the *Tech Notes* section that's centered on MPW® development, if that is the kind of information you're seeking). Once you find the needed information, you could share it with others and, as appropriate, put it onto the next version of the CD.

Apple wants to create links between more static information on the CD-ROM and more dynamic information from on-line systems.

This would create a kind of cybernetic feedback loop; the electronic-media entities we send you, such as the *Developer CD Series,* would get smarter at every pressing. It's not just that the disc will contain more information, but also that the feedback would make the information progressively more relevant to you. It creates a body of information that is self-defined for the intended audience and that can learn, so to speak. The system's real beauty is that the information it contains becomes more germane; it is increasingly based more on *what you need* than on *what we think you need.*

Building Successful Metaphors. To be truly successful, we must build common metaphors for the various kinds of electronic information. Because the Finder doesn't always extend well to information, we're experimenting with other ways of getting into interfaces. I want to stress that we aren't trying to replace the Finder; rather, we are trying to augment it to facilitate information location.

The magazine metaphor seems to make the most sense. The term *magazine* is actually shorthand for an *editorial context* and can be extended to include newspapers, journals, newsletters, and even the collected offerings of a publisher, movie studio, political party, or other organization with a distinct pattern of filtering and presenting information. If you subscribe to a magazine, you are essentially subscribing to a context of information in which you are interested. Other people who subscribe to that same magazine are likely to have at least some similar interest and therefore participate in that same context.

If a magazine is a focal point for desired information that can be browsed through and digested as desired, then by building a system that allows anyone or any group to do this, we will move from the era of desktop printing (which we call desktop publishing) to true desktop publishing—building electronic bridges based on interest and need between geographically separate people. And unlike traditional magazine publishing, which, except for letters to the editor, is a one-way enterprise, electronic magazines (dubbed "hyperzines" by Stuart Greene) can become an interactive forum for the exploration and development of ideas.

Electronic interactive magazines are an idea whose time has come. Right now, a HyperCard front end allows us to utilize rich, highly configurable multimedia components. As we build more effective front ends and tie the static "infomass" on your CD-ROM to on-line systems, you will be able to automatically search a variety of sources for specific information and automatically compile your own customized, individualized hyperzines.

Most of the benefits are obvious, but one in particular may not be readily apparent: By being able to collate information from various and sometimes totally unconnected sources, you may begin to make information connections that you didn't make before. You might combine what at first look like unrelated ideas, and—whammo—inspiration strikes: You've come up with the next great desktop-something software package. You've converted the information into something truly organic: knowledge and vision.

In the hyperzine metaphor, how could you actually get at the needed information? Agents, or guides, are one alternative. Semi-intelligent agents (very smart Find File applications) would locate specified information by perusing discs and, when appropriate, automatically and seamlessly dial into other forms of electronic media (such as AppleLink, CompuServe, USEnet, bulletin board systems, and other networks) to sift through the information and bring to you the significant pieces.

With a Macintosh, a CD-ROM drive, a modem, and a well-designed front end, an individual anywhere can essentially have the power of a globally focused knowledge-generating neural net. The idea is to create a place in which you can ramble through things, play, poke, examine, and learn—an electronic garden, a mini virtual reality in which individuals can make connections among things that they couldn't make before, to discover things they didn't realize were there. As the garden grows, raw data becomes information that then undergoes the metamorphosis into knowledge.

Can you say "Satori" Gumby?

GetNextEvent

The "" indicates the trade shows/events at which Apple Computer, Inc. will exhibit. This list may be incomplete. If you have information about a show that you want listed here, contact Diane Wilcox, 20525 Mariani Avenue, Mail Stop 75-3B, Cupertino, CA 95014, Link: WILCOX.DM. For further information check the Events folder on AppleLink (path: Developer Services: Events).

Date	Show	Location	Contact	Phone/Link
6/7-6/9	Macworld	**Melbourne, Australia**	POSTPONED 'TIL 1991	
6/12-6/15	AEC—Architectural, Engineering, Const.	**Atlanta, GA**	George Borkovich	215/444-9690
6/14-6/17	Macworld Asia	**Singapore**	Wei Sinclair	Link: FEG0002
6/19-6/21	PC Expo	**New York, NY**	H. A. Bruno, Inc.	201/569-8542
6/24-6/27	DAC—Design Automation	**Orlando, FL**	Marie Pistilli	303/530-4562
6/25-6/27	NECC—Nat'l Educ. Computing Conf.	**Nashville, TN**	Sharon Finke	503/686-3537
6/26-6/28	DEXPO East	**Boston, MA**	Expoconsul Int'l	609/987-9400
6/26-6/28	Seybold Digital World Conference	**Beverly Hills, CA**	Kathleen Kaiser	213/457-5850
7/17-7/19	Macintosh/New York	**New York, NY**	Cambridge Marketing	617/290-0400
7/20-7/211	A-2 Central	**Kansas City, MO**	Sally Dwyer	913/469-6502
7/29-8/1	NUL—Nat'l Urban League	**New York, NY**	Miriam Morales	212/310-9037
7/31-8/2	AAAI—Amer. Assn. for Artif. Intelligence	**Boston, MA**	Steve Taglio	415/328-3123
8/7-8/9	SIGGRAPH	**Dallas, TX**	Barbara Voss	212/752-0911
8/8-8/11	Macworld Expo	**Boston, MA**	Mitch Hall & Assoc.	617/361-8000
9/6-9/9	CyberArts International	**Los Angeles, CA**	Miller Freeman Expositions	415/995-2471
9/14-9/16	AppleFest '90	**San Francisco, CA**	Cambridge Marketing	617/290-0400
9/17-9/19	FCC—Federal Computer Conf.	**Washington, DC**	Information Development Corp.	301/961-8990
9/17-9/19	Presentations '90—Graphics & Multimed.	**Los Angeles, CA**	Cambridge Marketing	617/290-0400
9/19-9/22	Apple Expo	**Paris, France**	Fabienne Roch	Link: FRA.PROMO
10/3-10/5	Seybold—Electronic Publishing	**San Jose, CA**	Kathleen Kaiser	213/457-5850
10/4-10/6	Macworld Expo	**Sydney, Australia**	Macworld Expo	Link: AUST0261
10/18-10/20	Closing the Gap '90	**Minneapolis, MN**	Mary Ann Harty	612/248-3294
10/22-10/26	Systec—Computer-Integrated Manuf.	**Munich, Germany**	MMG	089-51070
10/25-10/30	Orgatec Koln Int'l Office Trade Fair	**Cologne, Germany**	Koln Messe GmbH	0221-8210
11/4-11/7	SCAMC—Symposium on Computer Applications in Medical Care	**Washington, DC**	Greg Thomas	202/994-4285
11/12-11/16	Comdex	**Las Vegas, NV**	Interface Group	617/449-6600
11/13-11/15	Autofact	**Detroit, MI**	SME	313/271-1500
12/4-12/6	Macintosh L.A.	**Los Angeles, CA**	Cambridge Marketing	617/290-0400
1/10-1/13	Macworld Expo	**San Francisco, CA**	Mitch Hall & Associates	617/361-8000
1/22-1/24	Uniforum	**Dallas, TX**	PEMCO	800/323-5155