The Developer Business Report *September 1996*

AppleDirections

CONTENTS

APPLE NEWS

Macworld Boston: New Developer Programs, Enhancements to Current Support Announced

At the recently concluded Macworld Boston, which included a celebration of the worldwide Macintosh customer base reaching 25 million, Apple Developer Relations (ADR) introduced several programs to help you market your products. ADR also announced a variety of enhancements to existing developer support programs to help make your development efforts less expensive and more successful.

See page 2 for important information about Apple's Mac OS 8 strategy

At the Worldwide Developers Conference in May, Apple Computer CEO and Chairman Gil Amelio and Apple Developer Relations Vice President Heidi Roizen committed to delivering new programs to enable your products to reach customersmore easily. Third-Party Marketing in ADR is rolling out a number of these programs as well as a variety of opportunities to help you increase awareness to your products within the Apple installed base and emerging markets.

please turn to page 7

STRATEGY MOSAIC

A Talk With Apple Developer Relations Vice President Heidi Roizen

New Directions for Apple's Developer Support Organization

By Paul Dreyfus, Apple Directions *staff*

On January 25 of this year, Heidi Roizen became the first vice president of Apple Developer Relations (ADR). A visible, vocal member of the developer community as founder and president of T/Maker, Heidi promised to bring a fresh perspective and new energy to the task of supporting Apple platform developers. Since then, she's spent a lot of time talking to people within Apple Computer and in the developer community to chart the course for her new organization.

I'm sure that many of you want to hear how ADR is going to work with you in the future, since it's a key part of your strategy. It's also an increasingly important part of Apple strategy. In the words of CEO and Chairman Gil Amelio, "Developers are the lifeblood of Apple platforms," and he's taken steps lately to be sure Apple is doing everything it can to work in partnership with developers. For example, Heidi Roizen reports directly to Chief Administrative Officer George Scalise in order to raise the visibility and importance of ADR within Apple. Also, at a time of cost-reduction measures throughout the company,

please turn to page 3

AppleDirections
Volume 4, Number 9

Apple Directions, the monthly developer newsletter of Apple Computer, Inc., communicates Apple's strategic, business, and technical directions to decision makers at development companies to help maximize their development dollar.

Editor
Paul Dreyfus (dreyfusp@apple.com)

Technical Editor
Gregg Williams (greggw@applelink.apple.com)

Business & Marketing Editor
Kris Newby (newby.k@applelink.apple.com)

Associate Editor
Anne Szabla (szable@applelink.apple.com)

Production Editor/Graphic Designer
Lisa Ferdinandsen (lisaferd@applelink.apple.com)

Contributors
Brian Bechtel, Peter Bickford, Alex Dosher, M. Raymond Jason, Caroline Rose

Production Manager
Diane Wilcox

Prep and Print
Consolidated Publications, Inc., Sunnyvale, CA

UPDATE

Amelio Announces New Operating-System Strategy

We received word of the shift in Apple's operating system release strategy after we went to press; its importance dictated that we report it to you as quickly as possible. As a result, we pulled the Editor's Note and included this report in its place. Below is what we know as of August 10, 1996; as soon as we know more, we'll provide additional details, both in future issues of Apple Directions *and in the news at the Developer World Web page (**http://devworld.apple.com/**).*

—the editors

• • •

During his keynote speech August 8, 1996, at Macworld Expo Boston, Apple Computer, Inc., President and CEO Gilbert Amelio described Apple's efforts to reorganize itself into a more successful company, talked about new products being shown at Macworld, and previewed some exciting new technologies that Apple is currently working on. As part of that speech, Dr. Amelio announced a fundamental shift in the way that Apple delivers new operating-system functionality.

Dr. Amelio stated in his keynote speech that Apple is changing its strategy to deliver new functionality through incremental releases rather than large monolithic releases. Moving forward, Apple intends to follow the industry model of shipping software releases in incremental segments. This release strategy will get new software developments out to the market more quickly and allow both developers and customers to take advantage of new software developments sooner.

The motivation for this change is that Apple believes that its current model of monolithic system-software releases isn't working, and that it doesn't allow Apple to get software advancements out to customers and developers soon enough. Therefore, Apple has decided to change its introduction strategy for system software to adopt the industry model of shipping releases in incremental segments more regularly.

In line with this new strategy, Apple now views Mac OS 8 not as a single large release but as a series of incremental releases. Apple recently shipped an incremental release, System 7.5.3, and plans to issue the next system software release in January 1997, followed by another release in July 1997. These releases should begin delivering some of the new features from Mac OS 8.

The planned January 1997 release includes some new functionality and integrates the latest versions of OpenDoc, Open Transport, QuickTime, and Cyberdog into the Mac OS. Apple intends to ship a developer seed of this release in early October. At present, no details about the contents of the July 1997 release are available.

Apple's new strategy for delivering operating-system functionality has caused some changes in previously announced plans. Apple has canceled the Mac OS 8 Developer Release: Compatibility Edition CD, which was scheduled for delivery to developers this month. The reason for this is that the concept of shipping this or any other large, monolithic developer release is not consistent with Apple's new software strategy of shipping incremental releases.

Apple believes that it needs to work closely with developers to ensure that this new strategy meets their needs and allows both Apple and them to offer Mac OS customers the latest software technologies and applications. Once Apple has fine-tuned this new strategy based on developer feedback, the company will share a more detailed strategy with developers and the public. ♣

A Talk With Heidi Roizen

continued from page 1

ADR's budget and number of employees are actually increasing this year.

The specifics of working with you, and aligning Apple's efforts with your needs, has been left to Heidi and ADR. Some of you have had the chance to meet with Heidi and hear, firsthand, how she views the task of developer relations. Many others of you have corresponded with her via e-mail; by her own admission, Heidi answers well over 100 e-mail messages per day. Now that she's had a good six months to adjust to Apple, to plot her new organization, and to formulate her vision, we thought it would be a good time to ask Heidi to share her thoughts with all of you. *Apple Directions* Technical Editor Gregg Williams and I met with her in early July; the following is our interview with her.

• • •

Apple Directions (AD): *You've had a chance to settle in and put an organization in place. Now that ADR is really up and running, how would you describe your vision for Apple developer relations?*

Heidi Roizen (Heidi): I'll start right off the bat by saying we never could do what we need to do in ADR without the commitment from the very top of the company—at the Gil level—which we have. Basically, in order to do some of these new things, we need money. We need people. We're getting those, too, so we're going to be able to do a lot of things we've never been able to do for the developer community.

So what's the mission of ADR? Number one, to work on the business and technical proposition that will give people the reasons they need to support Apple platforms. Number two, to act as the touchpoint between the development community and Apple.

Regarding number one, there's no amount of work ADR can do by itself to make a solid business proposition around our platforms. We can come up with new ideas, we can channel people in the right direction; but we, ourselves, cannot make a market with Apple's platforms. To do that, all of Apple has to have good business practices, competitive offerings at good prices with volume propositions, and all that kind of stuff.

"...we're going to be able to do a lot of things we've never been able to do for the developer community."

So, even though we can't invent the proposition on our own, we can formulate the messages around our business and technical propositions for developers, and communicate it. That's partly where the second part of the mission comes in.

The aspect of being the touchpoint between developers and Apple: It's really important for ADR to be the voice of the developer community inside Apple, because developers have really good ideas about what Apple ought to be doing. Also, since we depend on developers' support to move our platform forward, if they tell us they're not going to support something, we need to know that very early in the game.

How do we accomplish that? A big piece of it is just having the manpower and the money. Since the beginning of the year, we've added more than 50 people to Brian Gentile's Evangelism organization. David Krathwohl's international support efforts have also grown. We're expanding the Technical Services group under Gary Hornbuckle. We've started a new group under Bill Caswell, the Business Development group, which will be a small but highly focused effort to help find new business development opportunities. And in Jonathan Fader's Developer Marketing group, for the first time, we're not just marketing to developers, but we're also helping developers market to customers.

Right now, we're in a little bit of an awkward phase, where we know what we need to do but we haven't got the plans in place to do it yet. Apple still is a $10 billion company and you have to obey the rules of hiring, budgeting, and starting new projects. We can't just say, "OK, we need to do a channel campaign," and then start one tomorrow.

Also, we want to be sure to make the best use of each dollar we spend, not just in ADR, but Apple-wide. We have to ask, "Would we rather have a dollar go to some activity in ADR, or would we rather have the same dollar go to another Mac OS 8 engineer?" I need to be able to be sure that everything ADR does is the best investment for Apple to make on behalf of the developer community.

Maintaining Developer Loyalty

AD: *There's been a lot written about the challenge facing Apple in general, and Apple Developer Relations in particular, of maintaining developer loyalty in the face of all the problems Apple has faced in the market recently. How do you view the situation, and what can you do about it?*

Heidi: The problem is very simple. We need to make sure Apple platforms provide a healthy business opportunity. How do you make something a good business opportunity? You increase the chance for revenue. You decrease the cost of getting that revenue, and you try to eliminate risk.

Increasing revenue opportunity is a volume proposition, and, as I said above, that's something all of Apple has to work on. Through the work Apple and Mac OS licensees are doing, through Pippin and other new uses of Mac OS technologies, we're simply trying to make a bigger market. Bigger markets naturally attract more investment and provide more opportunity for developers to succeed.

The other side of the equation is cutting costs, which, if we can do it, also makes it

October *Apple Directions* Online

October's *Apple Directions* will be available by September 15 on the Web at
http://devworld.apple.com

more attractive for developers to support us. We can reduce their cost of development through better tools, porting assistance, engineering consulting teams, and a variety of other steps. We can make it cheaper for developers to reach customers with some of the channel programs we're initiating.

How do you work to eliminate risk? You make commitments to developers. You follow through on the commitments. You share your data with them. You share your forecasts with them. You say what you're going to do, and then do it, so the venture-capital community develops a greater trust of Apple.

AD: *How will you be able to determine whether these measures are working? If Apple and ADR are successful, what will the Apple developer community look like in the future?*
Heidi: Our success won't be measured in just the mass number of developers we'll have signed up to our programs. That's important, but we'll be looking at the value of their underlying businesses. We'll also be successful if we're gaining the support we need from developers in our key market segments.

I think it is a losing battle for Apple to continue to focus on overall market share; instead, we need to focus on those markets where we already have a strong position, like the Internet, media creation, publishing, learning—high-growth, high-opportunity areas. Part of Apple's focus on those areas will be working collaboratively with developers to make things happen in those markets.

You may even end up with fewer developers, but they'll be the developers who are aligned with us to go after those key market segments. The point is that it doesn't matter to me how many registered developers we have. What matters to me is how many great products we have and how well those products do in the market.

Supporting All Developers, Large and Small

AD: *Does that mean you'll be narrowing ADR's support efforts to working with a select few developers?*
Heidi: Absolutely not. No way. In fact, let me get on my soapbox for a minute and dispel that idea before it has a chance to go anywhere.

Lately in my e-mail I've been following an unofficial debate in the developer community about whether or not we should continue to support all developers, both the little guys and the big guys. The little guys say, "Support us because the big guys have the resources to support themselves," while the big guys argue that their strategic importance to Apple's markets makes it more important for us to support them.

The fact is that we need to support both big and small developers. Just because big developers have more resources, that doesn't mean they devote the optimal amount to Macintosh. And remember, often the sale of our platform to an individual or corporation hinges on the availability of a "validator" application. So, we need the "big guys," and we need to work with them. That said, it is also fairly safe to say that the bulk of the innovative, exciting products we see today come from the smaller companies. So, it would be very shortsighted not to help them out too.

This doesn't mean that we support everybody with everything, and certainly not that we treat all developers the same. It's interesting, though: In the feedback I receive, people's satisfaction with our developer support doesn't correlate to how large their company is.

AD: *In other words, proportionately speaking, there are as many small developers who complain about Apple as there are large developers?*
Heidi: I prefer to think that there are just as many satisfied developers among the big guys as among the small guys. My point is that out of the collective mass of developers—small and large—there are some who are going to create products that really move our platform forward. Since we have limited resources, we should be investing in those developers who, with us, can build successful mutual business opportunities. And I can think of many small developers who fit into that category; for example, think of the many one-, two-, or three-person shops out there that have gotten behind OpenDoc and Cyberdog. The big guys make splashy headlines when they announce OpenDoc container support, while the little guys help us achieve critical mass with the sheer volume of parts that they add to the picture.

I think there's a synergy out there among all developers. Sometimes a big developer leads the way with the large amount of resources they can use; other times it's the little developers who carry the torch because of their ability to focus and move quickly. Our challenge is to find the developers who can help us carry our platforms forward, and give them what they need.

Devising New Programs

AD: *So what you're really about is improving developer support for all developers?*
Heidi: That's right. How do we do that? The two biggest things are, first, putting more programs in place, and second, rebuilding the relationship with the development community. Regarding the first, we want to listen to developers and add programs that address the needs they have. Since I spent a lot more time being a developer than I spent being an Apple employee, I'm attuned to the sort of things Apple could do, that only Apple can do, that will make its platforms better from the perspective of the developer. We do have to keep our costs in mind; we can't go spending huge numbers of dollars on new programs, because we do have finite resources. We'll be looking at programs where the incremental costs of supporting developers is near zero. We'll also be looking at programs that are so beneficial to developers' profitability that they'll be willing to help defray the costs. We also need to be sure our programs are available to and benefit as many developers as possible.

For example, instead of charging developers for their information, we're going to give as much as we can to them for free on the Web. Same goes for software. Also, the more information we can put on the monthly Developer CD, the better off we're all going to be. These are things we can do where supporting more developers costs next to nothing, so of course we're going to do that.

For examples of new programs that developers might be willing to contribute to financially, let's talk about our test labs and the way we provide technical support. We know developers need to be able to ship Macintosh products more quickly. An extra week or two, or four, can equal many thousands of dollars in revenue. So what if we put together the world's best testing and certification labs where developers can test new products more quickly? Developers will see an economic benefit from this, because they'll be able to take products to market faster, so they might not mind sharing some of the cost with us.

Also, the only kind of technical support developers can get from Apple today takes place by e-mail, right? You can't get phone support. Yet many developers would be happy to pay 50 bucks an hour to talk to someone on the phone just to speed up their development process.

Developers themselves have said this to me. So that's another new idea we're looking at.

These types of programs go back to an idea I have that if it's expensive for us to offer a service, but it's valuable to developers, developers will vote with their dollars. If they're willing to contribute, we know it's important to them. If they're not, we just don't do it. I think we can do a much better job coming up with those kinds of programs, whether they be on the development side or the sales/channel side.

Channel Programs

AD: *Speaking of channel programs, we know it's a top priority for ADR to address the shelf-space problem and work with retailers and other sales channels to increase the visibility of Macintosh products. What's the latest on how you're addressing that?*

Heidi: We've historically been a minority player in the retail channel, which means we first have to be thinking, "OK, let's forget about the channel completely. Let's go on to the next thing." We could focus most of our efforts on catalog sales, where we've always been strong, and invest in new distribution channels, like the Web and virtual storefront CDs, and so on.

So the question becomes: Do you play catch-up, because it's the price of admission to be in this business? Or do you go ahead and do the next thing, which is riskier, but maybe gives a higher payoff because you're ahead of the competition? Right now, I believe we have to do both, which is part of the challenge.

Rebuilding the Relationship With Developers

AD: *You mentioned rebuilding the relationship Apple has had with developers. How do you go about doing that?*

Heidi: We need to build the concept that there is a person at Apple who gets up every morning wanting you to be successful as a developer. This is a tricky one, because out of the thousands of developers we need to be in touch with, there are probably about 500 companies we can support at any one time with a personal relationship. What will make this kind of relationship possible is that we won't always have to personally support the same 500 developers each month.

I look at it this way: Maybe you need some help this month. You've got a product that's about to ship and it's something Apple needs; as a result, you get more help. After it ships, maybe for six months, you can make use of existing programs that meet your needs, allowing us to focus our "individual attention" on someone else. It's a give and take.

It goes back to what I was saying before: We have to find the developers who serve our platforms' strategic interests the most closely, and then make sure they get what they need. One month, that might be one of the big "volume" players, the next it might be a "breadth" player who's not as large, the third month it might be a two-person "emerging" player. We're going to have to look at the developer community in terms of its segments, learn what each segment's needs are, and work to meet their needs.

"We're going to work to tailor our approach to the needs of different developers . . . we shouldn't be throwing the same blanket programs at everybody."

A large developer loves the idea of a huge installed base; the larger the customer base, the more profitable they can be. But let's look at the lone developer who just quit his job to start out on his own; his wife (or her husband) is saying, "If you don't make a profit within six months, you have to go back to work." They've only got 50,000 bucks saved up; that's the only money they can spend before they have to go get jobs again. If it costs $50,000 to develop a Macintosh product, or $100,000 to develop on Windows, they're going to develop on the Mac, case closed. So the dynamics and the metrics and the opportunities are very different, depending on the developer.

We're going to work to tailor our approach to the needs of different developers. Large-volume developers need to have one kind of relationship with us if they're going to stay on our platforms and be successful. The breadth developers or the emerging developers have different sets of needs; we shouldn't be throwing the same blanket programs at everybody.

Thinking Entrepreneurially

AD: *From what you're saying, a logical conclusion one could draw is that, a year from now, our programs will look quite different. Some things will be gone, and other things put in their place to better target the different segments of the developer community. How do you think things will look?*

Heidi: I don't have a precise picture in my head, but I know we're going to have to think very entrepreneurially and look at everything from the ground up. I think in the past at Apple, many things were done the way they were done just because they'd always been done that way. Instead, we're going to have to look at all of our support to be sure it's truly meeting an identified need, and that it's the best investment we can make on behalf of developers, compared not only to what the rest of ADR is doing, but also compared to everything else we're doing at Apple.

For example, in the past we've done some very extravagant marketing brochures and booklets for developers, about our services, our technologies, and other parts of our business. Did they really have to be that fancy? Did we really need to spend that kind of money on full-color production, artwork, and so on? I can't even imagine how much was spent on some of those pieces, especially when real information developers want might not even have been in them.

My bias: Developers don't want a fancy marketing piece that says, "Gee, we're a great company and we've always been an innovator." They know that already. They want market data, the real business proposition behind a technology, hard data they can use and solid programming advice and code. They don't care that much how it looks. So here's one of those cases where I would just say, "That was the old way of doing things." Apple has always meant top design and production values, but maybe that's not what our developers always want. We have to be willing to jump to a new way and to think: "If we were a startup, how would we do it? If we didn't have all the legacy money built in, and an established way of doing things, how would we get material out to the developer community?" I can almost guarantee it

wouldn't be through expensive, full-color brochures.

AD: *OK, now we're going to put you on the spot. Given that approach, do you think the paper version of* Apple Directions *will still exist a year from today?*
Heidi: I don't know. I think the content's really important. I just don't know if paper is important. One benefit it has is that I think *Apple Directions* looks nice and clean and all that, but obviously, you're not spending a fortune on the production values. I like that!

AD: *Thanks, We've never had a fortune to spend on it.*
Heidi: One of the things that I think we've been really pushing through the whole organization is the concept of separating the strategy from the content created to support the strategy. We also need to separate the mechanisms used to deliver the content from the content itself. You could start by working backwards from the concept of "I run a magazine. What should I put in it every month?" I think a more solid approach sounds like this: "My job is to keep developers up to speed on the latest technical information. What do they need? And how do I deliver it?"

Satisfying Developer Needs

AD: *How is the decision-making process going to happen? How, as an organization, will ADR make decisions on what to keep doing, what to change, what new things to do, and what to throw out?*
Heidi: What I've tried to do in the organizational structure is, instead of making people responsible for a particular product, I'm giving people responsibility for satisfying a particular aspect of developer needs. So instead of having Developer Press over here and a group that delivers the Worldwide Developers Conference over there, now we have Evangelism, Developer Business Development, Developer Technology Services, Developer Marketing, and International. All members of ADR have to understand the part they play in meeting developer needs so they can evaluate whether their particular product meets those needs.

It's both a top-down and a bottom-up process; working from the top, I'm pushing through the organization a new approach focusing on developers' needs rather than on legacy products. Now, I'm looking for the folks further down the management structure to push up and make the decisions that need to be made.

Take *develop* magazine; I'm not in the target audience. I'm not a programmer. I wouldn't read *develop.* Does that mean it's not important? No! But it means I'm not a person who is going to judge whether it's good or not.

So instead, I go to Gary Hornbuckle, who manages Technology Services, and I say, "Your job is to meet developers' technical needs.

"We have to find the developers who serve our platforms' strategic interests the most closely, and then make sure they get what they need."

You have to make it easy to be on our platform. Go! You know, that's your job. You figure out what that means."

And he might sit down and say, "Well, part of it's live support; part of it's a test lab; part of it's how-to technical information." So then he goes to some of the people in his group and says, "Your job is to deliver Apple technical concepts and sample code to the developer community."

Then that person can think, "Should that be in *develop* magazine? Should that be a CD? What should I create, to move us forward?"

In every case, I want the decision to be made on a very similar basis. Across the board, I want people asking, "How were developers served? How was their support of our platform affected by what I did? What is the incremental cost and benefit for what we just did?" And every year, we have to rank these projects and ask, "Are our developers getting value out of the money we're investing in each of these programs?"

Instilling Accountability

AD: *So nothing stays just because it's always been here. Everything is looked at from the ground up.*
Heidi: Everything has to be looked at from the ground up. In Apple's current financial situation, ADR is a cost center. I can't justify what we're doing based on some sort of profit that's going to come to Apple as a result. I have to justify it on more obscure measurements. But we have to be pretty comfortable that they're good measurements.

Just because we're not a profit center doesn't mean that we shouldn't run with an understanding of what Apple's getting for its dollars. And that, I think, is something that everyone in the company should be more attuned to. It's a bias I have, coming from a small company. Because in a small company, everybody's attuned, right? If you don't make your numbers for a couple months, you don't get a paycheck any more.

It's hard to instill that same sense of urgency—that same sense of accountability—in a company this size. Nolan Bushnell once said, "Money is America's way of keeping score." Even cost centers need to keep track of what they're doing, by assessing what Apple spends compared with what we—and developers—get for that money.

A New Era of Honesty

AD: *What have you been hearing from developers lately, and what is Apple doing in response to what they're saying?*
Heidi: The most important thing developers have said, right from the very beginning, is, "Tell us what you're going to do. Then do it. Don't tell us one thing, then do another." And I think we've had, in the last few months, a very good new era of honesty. The first step in rebuilding the relationship is ensuring that developers know that when we say "X," we mean "X." I think that's been hard for Apple to learn, but I'm very encouraged by the developer response. People are more inclined to trust and believe us, because they believe that we're actually telling it like it is.

We're also trying to bring developers more into decisions here at Apple. As a result, I'd say the second thing that's really changed is that developers feel the senior ranks at Apple understand the importance of the developer community, that they know the benefits of working together with them.

New Business Development Group

AD: *Could you say a little about what you've hired Bill Caswell to do, and what his Business Development group is going to be?*
Heidi: Think of Bill as a specialist. He's not the doctor you see first. You have to be referred to Bill. Let's say a developer, or someone in our Evangelism group, gets excited about some new product area. Or let's say someone at Apple investigates a new technology that, even though it shows a lot of promise, for one reason or another we have to give up. Bill is the guy you'd go to for help in building a business plan for that product or technology, to help get it funded so that someone else can be successful with it.

Since a lot of that funding will come from venture capital, part of his responsibility is understanding what venture capitalists are looking for. He'll also be sure that Apple is providing that information, both to the developer who's writing a business plan, and straight to the venture capitalists. A big part of his job will be to go out and form an active alliance with the venture community to boost the amount of business development that goes on for our platforms.

There is a more specialized role that his team will play, which will be to actively recruit developers we feel are strategic to Apple's needs. Let's say there's a developer who hasn't worked on our platforms before, but we need their products to make our own products successful. Bill will act as that developer's advocate, both internally and externally, trying to help that developer frame a business opportunity around our platforms.

The other really important area that Bill deals with is resolving cases in which we and a developer have products in the same space. Until now, what's frequently happened in these cases is that the Apple product manager—the very person who is in competition with the developer—has resolved the issue. That's not going to happen any more. Bill's organization will be an outside body—an advocacy group, if you will—that can take these issues and look at them objectively and try to solve them to everyone's mutual benefit.

Throwing Starfish Back in the Ocean

AD: *How would you sum up the work you've been able to accomplish so far?*
Heidi: Well, there's a story I like to tell about how I really view what we're up to, what we've done. There's a little girl on a beach that's covered with starfish. She's picking them up one at a time and throwing them back in the water. She gets to a spot near a man lying on the beach, and he says to her, "You know, little girl, there are so many starfish out there; you can't save them all. So, with all your efforts, what does it matter?" And she says as she picks up another starfish and throws it into the ocean, "Well, it just mattered to that starfish, didn't it?"

That's pretty much what we have to do here. We're probably not going to solve every developer's problems, satisfy everyone 100 percent, make everyone hugely successful. But if we align what we're doing closely enough with the needs of the various sizes and kinds of developers, and pick the right ones, we can make a difference to enough of them that we'll start to see dramatic improvements for the entire community. ♣

Editor's note: Heidi and all of Apple Developer Relations need your ideas to move forward. If you have any responses to what you've just read, specific or general, you can send your comments to Heidi directly at heidi@applelink.apple.com. She'll do her best to get back to you, but remember: She already gets over 100 e-mail messages per day. You might get a faster response by writing to me at dreyfusp@apple.com; I'll be sure to forward your ideas to the person at Apple who can do something with them.

APPLE NEWS

Macworld Boston

continued from page 1

The first comarketing initiatives, announced at Macworld, include access to Apple's list of customers to help your direct mail efforts; the availability of Apple collateral, merchandise, and product photography; and development of a comprehensive Macintosh solutions directory on the World Wide Web.

Additionally, Apple announced (or, in one case, reannounced) the following enhancements to its existing developer support programs:

- a new hardware leasing program that helps members of Apple developer programs stay current with new Apple hardware
- programming-level support from Apple's Developer Technical Support (DTS) engineers on a pay-per-question basis for Macintosh Associate Program, Newton Associate Program, and Apple Media Program members (this enhancement, first announced in May 1996, was reannounced so that all program members would be certain to know about it).
- a 50 percent price reduction on all books in the *Inside Macintosh* technical library, as well as the availability of virtually the entire library in HTML format at the Apple Developer World Web site (**http://devworld.apple.com/**)
- two new books for developers—one about Mac OS 8, the other about Cyberdog—and two new online courses from Developer University

Here is more information about each of these initiatives and enhancements.

New Comarketing Initiatives

ADR announced three new comarketing initiatives at Macworld. (More details about participation in these programs, along with a number of other comarketing opportunities, will be provided, on the Developer World Web site by the end of August.)

- *Access to Apple Customer Database.* Over the next year, Apple will give developers access to its list of customers from the Apple Customer Databases. Apple believes there is a great value in letting you inform our mutual customers about your Macintosh products. By the end of August, Apple will communicate the procedures you need to follow if you wish to use its list of customers for strategic direct marketing programs.
- *Apple marketing collateral.* To further assist your marketing efforts, you can begin ordering Apple collateral, videos, and merchandise from the StartingLine catalog. StartingLine offers up-to-date, customizable advertising and promotional materials to help make your sales activities, ad campaigns, and events more successful. You can find out more by sending an e-mail message to StartingLine at merch.tsr@applelink.apple.com.
- *Macintosh products online database.* To make it even easier for customers to find Macintosh software and hardware products, Apple

will be building the list of Macintosh third-party products currently available on the Developer World Web site into a truly comprehensive database. (New entries to this database are always included in It Shipped!, which appears on page 29 of this month's issue.) The database will also be available on the main Apple Web site (http://www.apple.com), and it will be given to Apple sales partners and resellers. We encourage you to include your product in the database if you haven't already done so; forms for submitting your product can be found on the Web at http://devworld.apple.com/mkt/thirdparty.html.

Hardware Lease Program for Apple Developer Program Members

The new Technology Refresh Finance Program gives Apple developer program members the ability to lease Apple hardware. This enhancement to the hardware purchase program is designed to help you gain affordable, immediate access to the very latest Apple hardware. The program offers a 12-month, no-down-payment financing option on selected models of high-end Apple hardware, letting you "rent" the systems with low monthly payments. The first payment is due 30 days following delivery.

"Our goal is to provide developers the upgrade path necessary to keep up-to-date with Apple's latest CPU technology, thus enabling them to design cutting-edge solutions for the Macintosh platform," said Jonathan Fader, director of Developer Marketing. "We are very pleased to announce this special program, which offers the industry's most creative technology refreshment and upgrade options. We firmly believe that working with the latest technology is a critical component of the developers' process."

The Technology Refresh Finance Program is available immediately to Apple developer program members in the United States (call 800-225-0513) and in Canada call 800-461-8159; contact Alain Page). Apple is working on making the program available in other countries, starting with the United Kingdom. Program features may vary according to local legal and commercial conditions.

DTS Support for Associates, Media Program Members

The Macintosh Associate Program, the Newton Associate Program, and the Apple Media Program (formerly the Apple Multimedia Program) now offer programming-level support from DTS engineers. This new support option, which has been available since May of this year, allows members of these programs to purchase support on a per-question basis. If you belong to any of these programs, you can begin to take advantage of DTS support immediately by submitting code-level programming questions by e-mail to devsupport@apple.com. DTS will respond to all questions within three business days. The charge per question is U.S. $50.

Inside Macintosh Less Expensive, More Accessible

In its continued effort to provide you with tools and technical resources at the most competitive prices, the *Apple Developer Catalog* recently announced a 50 percent price reduction to its full line of *Inside Macintosh* books. To find out about the new prices, visit Apple Developer Catalog Online on the Web at http://www.devcatalog.apple.com/; you can also order books from the online catalog.

Additionally, in response to your requests, virtually the entire *Inside Macintosh* library has been converted to HTML and posted at the Developer World Web page. You'll also be able to find *Inside Macintosh* in HTML on the September installment of the Developer CD Series, which is included with this issue of *Apple Directions* if you receive it as part of the monthly Developer Mailing. You can search all the volumes on the Web by using the site's searching capabilities or the Developer CD's Apple HTML Local Search application.

New Developer Books, DU Courses

Apple Press and Addison-Wesley introduced two new books at Macworld Boston to help your understanding of specific Macintosh development issues: *Mac OS 8 Revealed,* which gives an early look at the features of the next-generation version of the Mac OS; and *Cyberdog Programmer's Kit,* a book/CD package that provides the complete software development kit for Apple's compelling new OpenDoc-based Internet-access technology.

In addition, Developer University (DU) announced two new online courses. The first, called Introduction to Component Software, provides introductory-level instruction for OpenDoc programmers. The second, Game Development With Sprockets, is designed to speed your development efforts with the recently released Apple Game Sprockets. Both courses are available free of charge at the Apple Developer World Web site.

■ ■ ■

Apple and FORE Systems to Spur Mac OS and UNIX ATM Development

Here in the United States, it's hard to see the acronym *ATM* without thinking of automated teller machines—those automated kiosks that allow you to withdraw money from your bank account. In the networking industry, however, *ATM* stands for *Asynchronous Transfer Mode,* a networking and communications standard that can be used to create flexible, high-speed networking systems. ATM has several important advantages over existing networking systems:

- *Geographic scalability.* An ATM network can connect a few personal computers in a single building (on a local area network, or LAN) or a multitude of personal computers located anywhere in the world (using a wide area network, or WAN).
- *Bandwidth scalability.* Computer files are getting larger every day, and network managers are looking for easy ways to speed up their networks. A basic ATM network runs at a minimum of 25 Mbps (megabits per second) but can be upgraded to run on personal computers at up to 155 Mbps. Workstations and servers can run ATM as fast as 622 Mbps, and heavy-duty telephone company switching equipment can run at 2.2 Gbits/sec (gigabits per second) and beyond.
- *Guaranteed QoS (Quality of Service).* This is probably the most important feature to application developers. With ATM's QoS guarantee, an application can ask for either a given sustained bandwidth or a guaranteed data delivery rate. (The ATM network either agrees to the guarantee and delivers the connection to the specified destination, or it replies that it can't currently deliver such a connection.) ATM's QoS guarantee is a very attractive feature to developers who want to deliver media-rich solutions of various sorts over a network—for example, teleconferencing or broadcast-quality video playback and editing solutions—because an ATM network can deliv-

Why I Prefer Macintosh

In which *Apple Directions* readers answer the question "Why do you prefer to develop on/for the Macintosh computer instead of a Windows-based PC?"

From Bruce D. Kyle
Rogue Wave Software, Inc.
kyle@roguewave.com

Explaining why I love my Mac is as hard as explaining why I love my wife. I just do.

We fight, we make love, we play together, we grow apart and together again. And as the years go by I appreciate her obstinence, her orneriness and her affection. But mostly I admire her elegance.

I am the lone Mac-guy with about a hundred UNIX and Windows developers. They appreciate their code or the work they have done, but they never seem to love what they touch all day. Why?

I often explain to those who are trying to understand the idiosyncracies of the Mac, "You gotta love it."

From Steve Satre
FGM, Inc.
satre@fgm.com

A few days ago, while minding my own business and happily working on a Mac based mapping program developed by my company, I was assigned (temporarily, please!!) to another program based on Windows 95.

I needed a file developed by another employee in another state. He had a ThinkPad which he tried to use to create a floppy to mail the file to me (no e-mail available to him). The floppy drive wouldn't work on the ThinkPad . . . we think it's a driver problem. So, he FedEx'd the whole ThinkPad to me.

Thinking like a Mac person, I figured I could just hook up the ThinkPad to the network, select it on my PC, and pop the file over to my machine. Not that easy. Getting the machine configured on the network was excruciating. We got two PC experts within the company involved and, at one point, we had five people troubleshooting the problem. There goes the savings of buying a PC over a Mac! We finally solved the problem, but I was almost resigned to printing the file and typing it in on my PC. Give me back my Mac!

From Dave Wilson and Steve Wilson
Dave Wilson Personal Concepts
dwilson@best.com

Mac customers: Our new application, Let's Keep It Simple Spreadsheet (or Let's K.I.S.S.), is fundamentally different than other spreadsheets, and is based on dragging and dropping objects to carry out almost all operations. It also uses "fun" elements like color patterns, pictures, and sound effects. Since Mac users were the first to fall in love with a graphical user interface, it's easier for them to accept these major changes in the spreadsheet metaphor. We thought Mac customers would like the "fun" aspect of Let's K.I.S.S. more than Windows users.

Mac development tools: The Mac has a unique programming environment called *Prograph,* based on a dataflow visual programming language. Prograph gave us a competitive advantage in programmer productivity, enabling us to create a major productivity application in less than two person-years of effort.

Competition: The Mac software industry has fewer players than does Windows. We are therefore getting far more press exposure as a Mac product than we would otherwise get. The Mac market is a good place for underdogs. People seem to be cheering for us, just because we are willing to play "David and Steve against Goliath."

Love: We love the Mac. Our software is Mac-first so people can show it to their friends and say: "Let's K.I.S.S. doesn't run on Windows. If you want to use this cool software, you will have to buy a Mac".

• • •

Tell us why you prefer developing on/for the Macintosh computer instead of a Windows-based PC; your answers will be considered for use as part of this column. Send your contributions of approximately 250 words or less via e-mail to dreyfusp@apple.com.

er frames of video data just as they're needed.

Until now, it has been difficult to add ATM features to applications, and there was no easy way to bring ATM to the Mac OS desktop. Apple Computer, Inc., and FORE Systems recently made several announcements to remedy these shortcomings and make ATM-based customer solutions a new market for Mac OS and UNIX® developers.

Apple Releases ATM Middleware

Until now, it was difficult for ATM card vendors and application developers to bridge the hardware-software gap between the two in Mac OS computers. Apple has solved that problem by designing Apple ATM Middleware. This new software offers developers several advantages:

- If you have an application that uses AppleTalk or TCP/IP for networking, Apple ATM Middleware gives your application transparent access to the underlying ATM network. In other words, your application continues to work, but it doesn't take advantage of ATM features.
- If you use the ATM API in your application, then it will run on any card that uses the Apple ATM Middleware, and it will continue to run even if the user increases the speed of the ATM network or upgrades the Apple ATM Middleware. In addition, your application will take advantage of the ATM features written into it.
- If you are a vendor of ATM interface cards, you no longer have to write custom ATM middleware modules (such as modules for signaling, LAN emulation, or network management) to establish a "connection" between ATM-based software, the ATM card, and the ATM network. (The process of writing custom ATM middleware usually takes 12 to 18 months.) Instead, you need only write driver software for your ATM card to the ATM DLPI (data link protocol interface) that Apple ATM Middleware presents to hardware. The ATM DLPI is based on the UNIX DLPI standard. Because of this, vendors of PCI-bus ATM cards for UNIX systems may find a business opportunity in doing the modest amount of work needed to write a Mac OS driver and sell the same card to the Mac OS market.

Apple Agreement With FORE Systems

Apple has licensed the ATM API, which is conformant with the X/Open Consortium's specification, to FORE Systems, a leading vendor of high-performance ATM-based networking products. In return, FORE Systems has agreed to implement the Apple ATM API for the UNIX platform. This will enable you to create applications (for example, client applications that access very large databases through an ATM network) for both the Mac OS and UNIX platforms with one programming effort.

The ability to create ATM-savvy applications for the Mac OS will present significant opportunities for some developers. Today, customers in the broadcast and entertainment industries must use UNIX machines, for example, to retrieve and manipulate stock video footage stored on very large UNIX server machines. Many people in these industries already use Mac OS computers and value them for their flexibility and ease of use. They will be very interested in the ability to work on the same Mac OS computer they're already using instead of having to move to (and deal with) a UNIX-based computer.

The ATM Application Jump Start Program

Apple recently announced the ATM Application Jump Start Program, which provides all the software and hardware you need to start experimenting with ATM and writing ATM-savvy applications. Developers who qualify for this program will receive the following for a $7,500 licensing fee:

- a development license for Apple ATM Middleware
- a Whitetree WS3000 Ethernet/ATM Workgroup switch (needed to set up an ATM network)
- two ATM25 PCI cards (to connect two PCI-based Mac OS computers to the ATM network)
- limited direct support from Apple's ATM engineering team

Though a $7500 licensing fee may seem excessive, Apple is essentially licensing this technology for free, because the value of the bundled hardware (necessary to do basic ATM development) is about $8000. In addition, even if you decide against continuing your ATM development, you'll still get something useful out of your investment—you can use the Whitetree WS3000 switch within your Ethernet network.

To apply for the ATM Application Jump Start Program, send e-mail to atmdev@pop.apple.com, using "ATM Application Jump Start Program" in the e-mail title or subject line. Explain what your company does and why you want to participate in this program; be sure to include your company name, shipping address, fax number, and a contact name, e-mail address, and telephone number.

■ ■ ■

CI Labs Announces "Live Objects" Name for Validated OpenDoc Parts

Developers aren't everyday customers, and technologies aren't products.

Component Integration Laboratories (CI Labs), the nonprofit industry association that promotes, licenses, and validates multiplatform component software, has recognized these facts and has acted accordingly. At Macworld Boston, CI Labs announced that OpenDoc parts that have been validated by CI Labs can be identified with the trademarked name *Live Objects*.

The reasons for this new name are twofold. First, *OpenDoc* is the name of a technology and does not summarize to the average user the benefits of using OpenDoc parts. Second, since the OpenDoc technology promises that any group of parts, regardless of their origin, will work correctly together, it's important for CI Labs to ensure that parts are tested for their overall compatibility and that customers can easily identify quality parts.

Through the CI Labs Validation Program, part developers can ensure that their parts meet the quality standards that CI Labs has set up. Because of this, OpenDoc now has a significant advantage over competing component platforms—the OpenDoc software architecture and CI Labs' validation process work together to ensure a higher quality of component software than the other platforms can deliver.

Once a part has passed CI Labs' validation process, the part developer can advertise its part as a Live Object. From CI Labs, its members, and part developers, the public will learn the value of buying Live Object parts. In addition, the term itself will help customers visualize the advantages of using OpenDoc parts.

The Apple OpenDoc marketing team will begin using the term *Live Objects* as it begins marketing the advantages of OpenDoc-based

component software to the general public. (Apple will continue to use the term *OpenDoc* to promote the technology to developers.) IBM will use the term *Live Objects* to market the Windows, OS/2, and AIX versions of OpenDoc.

For more information about the validation program, see the CI Labs Web site at http://www.cilabs.org.

■ ■ ■

Mac OS Customers More Than Twice as Likely as Other PC Customers to Use the Internet

According to a recent survey conducted by the Georgia Institute of Technology, 28 percent of Internet users access the Internet from a Mac OS computer, up from 20 percent the last time the survey was taken, six months earlier. Since that's more than twice as high as the overall Mac OS computer share of the overall computer market, Macintosh customers are more than twice as likely as other personal computer users to connect to the Internet. The survey also indicates a disproportionately high usage of Mac OS–based Web servers; nearly 25 percent of webmasters surveyed said they used WebStar to set up servers, and 13 percent said they used MacHTTP. For the survey, Georgia Tech's Graphic, Visual, and Usability Center collected data from more than 11,700 Internet users in April 1996, updating the previous survey completed in October 1995.

At the same time, the percentage of Windows clients fell slightly; 58.6 percent of respondents said they used some version of Windows, compared with 61.5 percent who said they used Windows in the October 1995 survey. The survey is endorsed by the World Wide Web Consortium (W3C)—which exists to develop common standards for the evolution of the Web—as well as NCSA's Software Development Group (SDG), which develops Mosaic and other Web technologies.

For the the survey results, including demographic information about Internet users, see http://www.cc.gatech.edu/gvu/user_surveys/survey-04-1996/.

■ ■ ■

16 OpenDoc-Based Live Objects Make Their Debut at Macworld

Macintosh developers introduced 16 Live Objects—software products based on OpenDoc technology—at Macworld Boston and demonstrated many others throughout the show. One-quarter of the main Apple booth at the expo—approximately 6,000 square feet—was devoted to OpenDoc and many of the new Live Objects, including 30 stations staffed by Apple, IBM, and various third-party companies. These first shipping parts are important because they signal the change of OpenDoc from a technology to an actual market for products.

The Live Objects being introduced ranged from Digital Harbor's "work processor" called *WAV,* a new breed of desktop word processors, to Apple's Essential Live Objects Kit for Mac OS, a suite of OpenDoc components designed to speed your development of Live Objects.

Here is a list of Live Objects using OpenDoc technology that were introduced at Macworld. These products are scheduled for release in August and September:

- WAV, by Digital Harbor: "work processor" word-processing software that can be embedded into other OpenDoc containers to enable users to make full use of the Internet
- Virtual Field Trips in Geology, by Addison-Wesley New Media Products Group/West: an interactive multimedia exploration of fundamental geologic concepts for college students
- GeoInsight, by ComGrafix: a map insert part supporting map analytical tools
- C-Graph and C-Table, by Corda Technologies: data-graphing and table-editing software components that let users create charts and tables in Live Objects documents
- WorldWrite, from WorldSoft: a multilingual word processor that lets users add Live Objects to gain access to page-layout features, input capabilities, and other word-processing capabilities
- PageComposer!, from MetaMind: a part that combines drawing capability with Live Objects embedding to let users construct page-layout documents
- LEXI, from SoftLinc: a spelling checker, thesaurus, translation dictionary, and verb conjugator that supports English and 12 other languages
- PartFinder, from Kantara Development: a part that automatically finds Live Objects components at PartBank, Kantara's component software distribution Web site (http://www.partmerchant.com/)
- Outliner, from Eclipse Services: an outliner part that organizes interactive views of text, video, graphics, database, and Internet parts
- dtF Database Toolkit, from theta group: a part for creating dynamic, interactive documents with data from relational databases
- Sizzler from Totally Hip Software: a viewer part providing streaming graphics capabilities for Cyberdog; WebPainter, a painting and animation program, will include exporting to a Sizzler OpenDoc part
- Microbrew, from Network Multimedia: a high-level graphical software development tool
- WebBurst, from Power Productions: a Visual Java applet authoring tool
- Web Squirrel, from Eastgate Systems a Cyberdog part that builds spatial structures to keep track of Web pages, FTP sites, and other network resources
- AppMaker, from Bowers Development: programming tools to help you use OpenDoc technology
- StuffIt InstallerMaker 3.1 from Aladdin Systems: a part that lets you create Live Object installers

Other products and capabilities demonstrated at Macworld included the following:

- Live Objects were shown working compatibly with Microsoft's OLE, Netscape plug-ins, and Java applets.
- IBM demonstrated multiplatform capabilities with Live Objects source code on the Windows, OS/2, and AIX platforms.
- Corel demonstrated WordPerfect for the Macintosh with Live Objects container support.
- Claris showed ClarisWorks, which featured Live Objects components for connection to the Internet and interoperability with Java applets.
- Adrenaline Software showed the capabilities of Numbers and Charts, Live Objects spreadsheets and 3D charts with links to the Internet.
- CNET Direct demonstrated Buy-Direct.Com, a new Web-based software distribution service that will offer customers the opportunity to purchase traditional software as well as new component software. Customers can buy Macintosh Live Objects

and, in the near future, Windows Live Objects directly from independent software vendors. BuyDirect.Com can be found on the Web at http://www.buydirect.com.

Also announced was the Apple Live Objects Essentials Kit for Mac OS, formerly released in alpha form under the code name *KickStart* on the OpenDoc Developer Release 5 CD. The Live Objects Essentials Kit is a suite of multimedia OpenDoc components that Apple is providing to help you get started with your own OpenDoc software development. Currently available are 1.0 versions of the Apple's QuickDraw 3DMF Viewer and the Apple Button, which can embed sounds, speech and Web site addresses (URLs). Final 1.0 versions of other components are scheduled to be available soon, including QuickTime and QuickTime VR components; viewers for PICT, GIF, JPEG and TIFF files; and editors for basic text processing, drawing, sound, and text annotation. You can find the components at Apple's OpenDoc Web site (http://www.opendoc.apple.com).

■ ■ ■

New Macintosh Systems Reach 200 MHz; Apple Ships First Multiprocessor Macintosh Systems, Minitower Performa Systems

On the heels of its much-better-than-expected results for the quarter that ended in June 1996, Apple Computer released a variety of new Macintosh hardware systems. The new systems, all based on PowerPC 603e and 604e microprocessors, include a 200-MHz Power Macintosh system, the first Apple-branded Macintosh system to utilize multiple processors, and Apple's first minitowers designed specifically for the home.

Apple's latest hardware offerings are expected to match up in the marketplace against Intel-based systems. In the May 27 issue of *Microprocessor Report,* Editor-in-Chief Linley Gwennap said, "Surpassing expectations, Motorola and IBM have announced they are now sampling PowerPC 603e and 604e processors at speeds up to 200 MHz. At these speeds, the 603e should match up well with Intel's forthcoming 200-MHz Pentium, while the 604e takes on Intel's 200-MHz Pentium Pro."

Following is more information about the new systems. More complete product details are available in the press releases about the systems, which can be found at at http://product.info.apple.com/pr/library/1996/august.html.

Performance Boosts in Power Macintosh Line

New to the Power Macintosh line are the PowerPC 604e–based Power Macintosh 9500/200 computer, running at 200 MHz, for customers in high-end publishing, multimedia applications, scientific and technical markets, and Internet authoring. Also introduced were the PowerPC 604e–based Power Macintosh 8500/180 computer, offering 180-MHz performance, and the 150–MHz Power Macintosh 8500/150 computer, both of which are for in-house publishing, media authoring, and technical applications. The line now includes the new Power Macintosh 7600/132 computer, which received a performance boost to 132 MHz from 120 MHz. The new Power Macintosh systems are currently available in limited quantity; worldwide full availability for all systems is expected by September.

First Multiprocessor Apple Macintosh Systems

Responding to demands from the areas of graphics, video, Web authoring, 3D rendering, and CAD/CAM, Apple introduced an affordable multiprocessor configuration of its high-end Power Macintosh systems—the Power Macintosh 9500/180MP computer. The system features a card with dual 180-MHz PowerPC 604e chips fitting into the system's processor upgrade slot. Apple expects other manufacturers to develop multiprocessor upgrade cards for customers of Power Macintosh or Mac OS–based systems.

DayStar Digital previously released its Genesis MP Mac OS–compatible multiprocessor workstation. "Apple's entry into the multiprocessing arena will further strengthen the Macintosh as the dominant platform in areas such as graphics, publishing, video editing, and Web page authoring," said Andrew Lewis, president of DayStar Digital.

To support development of applications that take advantage of multiple processors, Apple introduced the Apple MP application programming interface (API) earlier this year. Developers who have committed to MP versions of their applications for Power Macintosh include Adobe Systems, Bungie Software, DayStar Digital, Deneba Software, Electric Image, Live Picture, MacroMedia, MetaTools, Metrowerks, Specular International, and Strata.

Apple's Power Macintosh 9500/180MP computer is expected to be available in most regions worldwide by September.

Minitower Performa Systems for Home Users

Again in answer to customer demand, this time coming from home users who need higher performance systems for Internet access, multimedia computing, and other sophisticated applications, Apple introduced the Macintosh Performa 6400 series of computers. The 180 MHz and 200 MHz PowerPC 603e–based Performa 6400 models—Apple's first minitowers designed specifically for the home—are equipped to meet the increasingly sophisticated needs of families, home office workers, and college students with the best value in home computing.

The Performa 6400 series includes two industry-standard 7-inch PCI slots that make it easy to add special capabilities, such as Avid Cinema, a digital video editing system from Apple that lets customers store and manipulate video images on their computers; the Apple PC Compatibility Card, for customers who need to run DOS or Windows-based applications; or a video card, for adding a second monitor. Prices for the Performa 6400 models, which ship complete with monitor, keyboard, and either 1.6- or 2.4-gigabyte hard drives, start at U.S. $2,399 and are available immediately through Apple resellers, although price and availability may vary depending on location.

■ ■ ■

Apple Cosponsors New Marketing Vehicle With Macworld Magazine

Macworld magazine's new *Web Explorer Virtual Expo* CD, set to debut in the magazine's December 1996 issue, provides a new way to market your products. Cosponsored by Apple

Developer Relations, the CD will include the Macintosh Product Registry, a Web-based list maintained by Apple Computer. You'll want to be sure your product is part of the registry, which you can do for no charge at the Apple Third-Party Product Web site; you'll also want to consider taking advantage of the other marketing opportunities provided by the *Web Explorer Virtual Expo* CD. Depending on how much exposure you want on the CD, and how much you want to spend, you can choose from these three options:

- full-level booth, with a custom-designed 90-second video of your product and company, product demo, detailed product information, and a link from the Macworld CD Web site to your home page (special charter price: $10,400, plus $1,000 per link to your Web site)
- junior booth, with a product demo, fact sheets, and a link to your Web site (special charter price: $6,800, plus $1,000 per link to your Web site)
- virtual station, with product fact sheets, your logo, a photo of your product, and link to your Web site (special charter price: $3,600, plus $1,000 per link to your Web site)

Macworld will do all the creative work, including the video production, voice-over, and script writing, as well as the design of the selling environment on the CD. All you need to do is provide the pieces and approve it. The deadline for purchasing booths and stations has already passed for the December 1996 debut issue. If you're able to enter your product in Apple's online Macintosh Product Registry at **http://devworld.apple.com/mkt/thirdparty.html** by August 20, it will be included in the registry on the December CD.

You've got plenty of time to sign up for space on volume 2 of the *Virtual Expo* CD, which will be included in the May 1997 issue of *Macworld.* You have until November 1 of this year to reserve a full-level booth and until November 30 for junior booths and virtual stations.

For more information about the *Web Explorer Virtual Expo* CD, and to purchase space on it, contact Cherie La-France at *Macworld* (408-688-0707 or cherie_lafrance @macworld.com). In the meantime, hurry and enter your product in the Macintosh Product Registry.

■ ■ ■

QuickDraw 3D 1.5 Announced; Developer Support Strong

Apple chose the SIGGRAPH conference in New Orleans to announce that a new version of its cross-platform 3D graphics toolkit—QuickDraw 3D 1.5—will be available by the end of September.

The QuickDraw 3D API lets you easily build 3D interaction into applications; many of your colleagues are making their products stand out in the marketplace by using QuickDraw 3D in innovative ways. QuickDraw 3D has also attracted a number of new developers to the Macintosh platform, including NewTek, Positron Publishing, and Vertigo Technologies. Dozens of products using QuickDraw 3D are expected to ship concurrently with the new version. To differentiate your product by giving customers easy access to 3D graphics, we think you'll want to start using QuickDraw 3D, too, if you haven't already.

With the new release, the QuickDraw 3D API, which provides real-time, interactive rendering for simple 3D models, will be available for the Mac OS, Windows 95, and Windows NT. QuickDraw 3D RAVE (Rendering Acceleration Virtual Engine) version 1.1 is currently available for the Mac OS, Windows 95, and Windows NT platforms. Introduced in February, QuickDraw 3D RAVE is the foundation technology used in QuickDraw 3D; it provides ultra-fast, real-time, workstation-quality 3D graphics on both Power Macintosh computers and PCs.

Among the companies shipping products to coincide with the release of QuickDraw 3D 1.5 is Electric Image, developer of the ElectricImage Animation System. "Our product combines the power of the fastest renderer anywhere with seamless support for QuickTime output for professionals working in both film and broadcast video production," says Jay Roth, CEO of Electric Image. "We will soon enable 3D acceleration through Apple's QuickDraw 3D RAVE that will let Power Macintosh users enjoy the same ultra-fast production that was before only available on Silicon Graphics workstations."

The following is a list of other applications using QuickDraw 3D that are either currently shipping or scheduled to ship at the same time as QuickDraw 3D 1.5.

3D Design/Animation: Artifice—Design-Workshop; auto • des • sys—form • Z; Electric Café—ModelShop; Electric Image—Electric Image Animation System; Fractal Design—RayDream Designer; GraphiSoft—ArchiCAD; Graphsoft—MiniCAD; InterStudio—Domus.CAD; MacroMedia—Extreme 3D; MicroSpot—3D World; NewTek—Lightwave 3D; Positron Publishing—MeshPaint 3D; Specular—Infini-D; Specular—LogoMotion; Strata—StudioPRO; Vertigo Technologies—Vertigo; Yonowat—Amapi.

Multimedia: PowerProduction Software—Digital Box Office; Linker Systems—Animation Stand; MacroMedia—Director 5 xtra.

Graphics & Publishing: MultiMedia Marketing—HoloDozo: Deneba Software—Canvas 5; Canto Software GmbH—Cumulus PowerPro

Games: Reality Bytes—Havoc; MacPlay—VR Soccer; MacPlay—Descent II; MacPlay—Virtual Pool.

Hardware Acceleration: Matrox—Millenium (PCI); Newer Technologies Radius—Thunder 3D(PCI); ATI Technologies—RAGE(PCI).

3D Clip Art: Plastic Thought—3-d Active; Syndesis—3D-ROM; Model Masters—Model Monger; Viewpoint DataLabs—3D Datasets; Acuris—Air, Sea & Land.

Miscellaneous: LightWorks—Photorealistic Renderer; Kandu Software—CADMover; Syndesis—InterChange (Windows); Spatial Technologies—ACIS Modeling Libraries.

For more information about QuickDraw 3D, read the *Apple Directions* article in the October 1995 issue, which can be found on the Web at **http://dev.info.apple.com/appledirections/oct95/quickdraw3d.html**. Or, visit Apple's QuickDraw 3D home page at **http://www.quickdraw3d.apple.com**.

Game developers interested in QuickDraw 3D RAVE should contact Mark Gavini, Macintosh games evangelist, at gavini@apple.com. Other software or hardware vendors interested in QuickDraw 3D should contact Shawn Hopwood, QuickDraw 3D evangelist, at s.hopwood@applelink.apple.com.♣

Technology

develop Issue 27: Listen Up!

Issue 27 of *develop,* Apple's award-winning technical journal, will tell you how to turn your application into a good listener. You'll also learn about OpenDoc, Apple Guide, Mac OS 8 assistants, and much more.

• "The Speech Recognition Manager Revealed" and "Adding Speech Recognition to an Application Framework"—With these two articles, you'll have your application recognizing speech in no time. The first is an introduction to the long-awaited API for speech recognition, and the second is an example of adding basic speech recognition capabilities to a PowerPlant application.

• "Working With OpenDoc Part Kinds"—Part kinds are like file types, only more so, and the choices you make about which part kinds to support will have a profound effect on users' experiences with your part editor.

• "Using Apple Guide 2.1 With OpenDoc"—Apple Guide 2.1 has been extended to work well in OpenDoc's brave new world of compound documents and processes within processes. Here's a look at the new features and how to take advantage of them.

• "Mac OS 8 Assistants in System 7 Applications"—Assistants will provide interview-based help in Mac OS 8, guiding users through complex tasks. This article gives some tips on designing an assistant and shows how you can implement one now, under System 7.

• "Game Controls for QuickDraw 3D"—First-person 3D applications, whether games or 3D modeling systems, need to constantly move the camera to reflect the changing point of view of the player. You too can inflict vertigo on your users.

The columns in this issue of *develop* will give you the scoop on the new LaserWriter driver version 8.4 and a library for traversing QuickDraw GX paths, as well as on how to

please turn to page 23

CD HIGHLIGHTS

Reference Library Edition, September 1996

This month brings a new version of our prototype Apple HTML Local Search application, and a wide selection of *Inside Macintosh, Macintosh Technical Q&As*, and several other books in HTML format for you to test it with. HTML appears to be the Next Big Format for our technical documentation, and HTML Local Search is how we'll be providing searching capability for local files, so now is the time to try them out and let us know what you think, using the survey on the CD; it will be seriously passé to complain about them once we've officially adopted them.

In addition to updates to the Macintosh Technical Notes and the Gestalt Selectors List, here are this month's new and revised items.

Apple Grayscale Appearance

The Apple Grayscale Appearance for System 7.5 specification provides the information you need to create a grayscale appearance for System 7.5 products. It consists of an Adobe™ Acrobat document and two folders of SimpleText files of enlarged graphics showing the details of human interface elements.

This version of Apple Grayscale Appearance for System 7.5 contains an updated appearance for the disabled state of all windows, dialog boxes, and controls. Because developers and users have reported how difficult it is to distinguish active windows from inactive windows, Apple has changed the appearance of all inactive windows and disabled controls. This updated appearance gives users better feedback about inactive windows and dialog boxes. It also draws attention to active windows and enabled controls.

This version of the document also contains the appearance for tab panels (tabs and content panes) that appear in dialog boxes. It shows tabs in small and large sizes.

Developer Notes

This month features a new note about the LaserWriter 12/640 PS printer, a new member of the Apple LaserWriter printer family. This developer note describes the features and capabilities of the printer and is aimed at software and hardware developers.

Inside Macintosh

Inside Macintosh is a collection of books, organized by topic, that describe the system software of Macintosh computers. Together, these books provide the essential reference for programmers, designers, and engineers creating applications for the Macintosh family of computers.

This month, *Inside Macintosh* is provided in HTML format for the purposes of testing the Apple HTML Local Search prototype search application. Some hyperlinks may be "broken," and some may connect to other Web servers on the Internet. To begin browsing, open the file !InsideMacMainPage.html with your Web browser.

The following books are included in addition to the *Inside Macintosh* series:

Apple Game Sprockets Guide
Apple Guide Complete
AppleScript Finder Guide
AppleScript Language Guide
Cyberdog Programmer's Kit

The *Inside Macintosh: QuickDraw GX* series is available online, and will return to the Reference Library Edition next quarter.

please turn to page 23

FEATURE

The Year 2000: No Big Deal, or Apocalypse Never

By Brian Bechtel,
Apple Developer Support

Why the Mac OS Platform Has No Problem With the Year 2000 and Beyond

There has been much speculation recently about various computers and their handling of the year 2000. The Mac OS and Apple Macintosh computers, however, do not have problems with the year 2000.

Q. Will the Mac OS handle the year 2000 correctly?

A. Yes. All Mac OS date and time utilities have correctly handled the year 2000 since the introduction of the Macintosh computer. The original date and time utilities (introduced with the original Macintosh 128K computer in 1984) used a long word to store seconds, starting at January 1, 1904. This approach means that the last date that can be represented correctly is 6:28:15 A.M. on February 6, 2040. The current date and time utilities, documented in *Inside Macintosh: Operating System Utilities,* use a 64-bit signed value, which covers dates from 30,081 B.C. to 29,940 A.D. For further reference, see *Inside Macintosh: Operating System Utilities.*

All localized versions of the Mac OS include Gregorian calendar support. In addition, the Arabic version of the Mac OS supports both the Arabic astronomical lunar calendar and the Arabic civil lunar calendar. The Hebrew version of the Mac OS supports the Jewish calendar. The Persian version of the Mac OS supports the Iranian national calendar.

Q. Will Macintosh software handle the year 2000?

A. In general, yes. The only problems we've encountered have been with developers who used their own date and time utility packages instead of the Toolbox calls supplied by the Mac OS. If you're using your own date and time utilities, you should test your software carefully to ensure that it handles all of the subtle issues of date and time conversion.

Q. What about the year 2019?

A. The Date & Time control panel constrains user input to dates between January 1, 1920, and December 31, 2019. This feature was added for compatibility with the original Macintosh System 6 General control panel, which limited dates so that there would be no ambiguity about a two-digit year (which was all that was normally displayed on a U.S. system). The Date & Time control panel uses the Script Manager function ToggleDate. In 1989, Apple enhanced ToggleDate to include an option that limits dates to the 1920–2019 range so that ToggleDate could be used by the General control panel and other control panels that had a date widget that operated in the same way. You can set a date beyond 2019, up to the year 2040, by using the Macintosh Toolbox function SetDateTime.

Q. Does February 29, 2000, exist?

A. Yes, according to the rules governing leap years, there will be a February 29, 2000. The Gregorian calendar is a solar calendar that measures time from the year of the birth of Jesus Christ. It contains 11 months with fixed lengths of 30 or 31 days and a twelfth month of 28 days that is lengthened to 29 days every fourth year (called a leap year).

The Gregorian calendar is a refinement of the Julian calendar, established in ancient Rome. The Julian calendar assumed a year of 365.25 days. The actual length of the solar year, however, is 365.2422 days, which produces an error of one day every 128 years. In 1582, Pope Gregory XIII adjusted the error that had accumulated over the centuries by canceling the ten days between October 5 and October 15. He further ordained that years divisible by 100 would not be leap years unless they were also divisible by 400. (From this rule, you can see that 1800 and 1900 are not leap years, but 2000 is.) This adjusted the Gregorian calendar to 365.2425 days and reduced its error to one day in 3,323 years. Many non-Catholic cultures adopted the Gregorian calendar over the succeeding centuries, calling Julian dates *Old Style* and Gregorian dates *New Style.* This change occurred in the United Kingdom and its colonies in 1752.

For More Information

For information on the Mac OS utilities for date and time manipulation, see Chapter 4, "Date, Time, and Measurement Utilities," of *Inside Macintosh: Operating System Utilities.* For information about other calendar systems supported by the Mac OS, see Chapter 9, "Localized Macintosh System Software," and Chapter 11, "Calendars and Dates," of *Guide to Macintosh Software Localization.*

please turn to page 17

References

• Apple Computer, Inc. *Inside Macintosh: Operating System Utilities.* Reading, Mass.: Addison-Wesley, 1994.

• Apple Computer, Inc. *Guide to Macintosh Software Localization.* Reading, Mass.: Addison-Wesley, 1992.

• Coyne, G.V. , M.A. Hoskin, and O. Pedersen, eds. *Gregorian Reform of the Calendar: Proceedings of the Vatican Conference to Commemorate its 400th Anniversary.* Vatican City: Pontifical Academy of Sciences, Specolo Vaticano, 1983.

• Royal Greenwich Observatory. *Information Leaflet No. 48: Leap Years.* Cambridge: Royal Greenwich Observatory. (You can contact the Royal Greenwich Observatory at Madingley Road, Cambridge, CB3 0EZ, United Kingdom, or you can obtain this leaflet from the Web at http://www.ast.cam.ac.uk/pubinfo/leaflets/leapyear/leapyear.html.)

• United Kingdom. Parliament. *An Act for Regulating the Commencement of the Year; and for Correcting the Calendar Now in Use.* 24 Geo. 2. c. 23, A.D. 1751. Anno vicesimo quarto GEORGII II. CAP. XXIII.

• Pope Gregory XIII. *Inter Gravissimus,* 1582.

HUMAN INTERFACE

Master and Servant

By Peter Bickford

The truth is simply too shocking to keep secret. Oh sure, I could make excuses for my actions: I could say that I was pressured into it, or that the needs of the situation demanded it. Still, it was all I could do to make myself even contemplate the terrible thing I was about to do. In the end, however, I decided that it was better just to do it and get it over with.

A couple of months ago, I bought a Pentium. A clone. With Windows 95.

It was to be the eighth computer in my house, joining the platoon of PowerBook and desktop Macintosh computers that Carolyn and I use in our offices or my music studio. We even have an old Macintosh IIcx that serves as the router in our home network. I've worked on any number of machines in my time—from CDC Cybers to Texas Instruments TI/99s. Whenever I've had to spend my own money for a computer, however, the one I brought home always came emblazoned with a six-color fruit logo. Still, I was about to begin a cross-platform development project in my spare time, and it seemed like a good idea to experience how the other half (OK, somewhat more than half) lives.

With your indulgence, let me share the story of one man's journey into the world of "real computers." As Nietzsche wrote, "That which does not kill me makes me stronger." Having lived the life of a Wintel user for some two months now, I am happy to report that I feel much stronger as a person. I have also scored over 9 million points on the Windows 95 bundled pinball game. And the cross-platform development project? Well, did I mention I was getting very good at the Windows pinball game?

A Clone of My Own . . .

If you think sorting out the Macintosh model numbers is confusing, you won't believe what it takes to buy a PC. There, the world divides into two basic parts: the "name brand" lines from companies such as Dell, Compaq, and HP, and the "clones," which innumerable storefront companies produce by assembling stock components into pretty much any configuration you want. In this way, the technical elite have a choice of going to a clone vendor and ordering a "Pentium 166/32 MB w/256K Burst, 1 MB DRAM Video, Triton Chipset, 2.1 GB IDE HD, 104 Key Kybd, 6X-CD-ROM, 2 Ser/1 Par, Mouse, Soundblast w/Spkr, 15" (13.4" LVA) .28 Monitor" for $1899, while the less geeklike "home buyers" can simply order the "Packard-Bell Invectra *N*," where the model number *N* is an apparently random integer between 1000 and 32,627. If you don't know which value of *N* to choose, simply look for the computer bearing a little white card that helpfully states, "Invectra 3000: Pentium 166/32 MB w/256K Burst, 1 MB DRAM Video, Triton Chipset, 2.1 GB IDE HD, 104 Key Kybd, 6X-CD-ROM, 2 Ser/1 Par, Mouse, Soundblast w/Spkr, 15" (13.4" LVA) .28 Monitor—$2199."

Wanting to save some money (and understanding the terms, since I studied Computer Science in college), I opted for a sizzling-fast 166-MHz clone. I plunked down my money and was told to come back in a week, after which time my new machine would be available for me. When I returned, I was handed a stack of hand-marked "driver disks," along with a half dozen cartons that had once held my processor, main logic board, CD-ROM drive, and so on, along with various leaflets describing their technical specifications. The salesman then handed me the big metal CPU chassis and congratulated me on my purchase. "All you've gotta do now," he said, "is plug it in, install Windows 95, and you'll be ready to roll." I hadn't actually bought Windows 95 with my computer since a friend of mine at Microsoft had been nice enough to get me a copy of it last year from their company store.

"Don't You Know You're Not Supposed to Touch It?!"

When I got home, I plugged everything in, turned it on, and got ready to bask in the geeklike glow of this new machine. The screen display appeared, incomprehensible "driver loaded" messages whizzed past, and the preinstalled Windows 3.1 loaded. Everything worked fine as far as I could see, except that the "Plug & Play" mouse didn't appear to be working. After an hour of futzing with the cables and mouse drivers, and even swapping in a different mouse, I was having no luck at all. Calls to the manufacturer's technical support line ended with the suggestion that I "check my system for interrupt and device conflicts." I attempted to do so for another hour before deciding to punt on the problem and just install Windows 95. After all, one of the major improvements in Windows 95 was the "Plug & Play" architecture, which was designed to handle just such a situation. The idea was that Windows would automatically detect my system's hardware, then configure all the various drivers automatically.

The Windows 95 installation itself was reasonably smooth, if a bit lengthy. Nearly an hour after I began, it had finished installing and was rebooting my machine to finish the configuration. When the machine came back up, it flashed a message that it had detected new hardware (a PS/2 mouse) and was installing the software for it. "Victory!" I thought . . . until I realized that I didn't have a PS/2 Mouse—I had a regular one, which *still* wasn't working.

I was up until 2:00 A.M. trying to get the computer to find its mouse. I lost a couple of hours when I discovered the truly cool Windows 95 pinball game, which, as a bonus, could be played solely through the keyboard. The only other downside was that the sound would cut out suddenly after a few minutes of play.

The next day, I gave up in frustration, packed the whole thing back up into my car, and brought it back to the store, saying, "I bought this yesterday and the mouse doesn't work. Fix it." They did, three days later, mumbling something about old drivers, interrupt conflicts, and phantom mouse ports on that particular logic board. They didn't have any suggestions on my sound problems, other than to hint that "on-board sound isn't the most reliable thing in the world." After many more wasted hours (and a growing addiction to the Windows pinball game), I eventually solved this problem by spending another $149 to buy a real Sound Blaster sound card instead of the "Sound Blaster–compatible" version that was built into my computer's main logic board.

By the time my machine was working properly, I'd learned one of the most important lessons of Wintel computing: fear of the system. My Windows-savvy friends emphasized this point: If the system appears to be working properly, leave it the heck alone. Or, as one put it, "Don't you know you're not supposed to touch it?!" Even the salesman who had sold me the computer made a big speech about how their stuff always works great—until some user monkeys around with the settings. Apparently, customers who get too curious about their machines' WIN.INI and other configuration files are a major source of technical support revenue for them.

User Control?

Two nights ago, the computer ate part of my development project. Yes, I know how stupid that sounds, and I haven't said anything like this about a computer since I was 13. It's the fact nonetheless. The disk file containing the main project window simply disappeared and can no longer be found. It's not in the Recycling Bin, and I've had no cause to move it. I received no messages asking me if I wanted to delete it.

I can recreate the work easily enough, but the incident just adds to a feeling I have—one I suspect many Wintel computer users have—that we're not really the ones in control here. Unless we learn to speak in the computer's language, work within "the rules," and stay on the well-lighted path, we can't even buy one of these machines, let alone get any useful work done with it. Moreover, there are potential hazards lurking everywhere in the computing environment. Load the wrong driver setup, or peek into that WIN.INI file without being careful, and the computer will do to your work what Brothers Grimm fairy tale villains used to do to recalcitrant children. Tread carefully, for here be there tygers.

The better Wintel developers realize that what they are often doing is creating a safe path through what can be a dangerous underlying system. Thus you see a great emphasis in that environment on "wizards" that guide users step by modal step through operations that are thought to be too complex or dangerous for users to do by themselves. Wizards are used for virtually everything, from installing software and troubleshooting device conflicts to building charts in Excel. The lack of freedom and control the user feels while using one of these wizards is thought to be a small price to pay for the fact that without them, many users would not be able to accomplish these tasks at all.

Apparently, customers who get too curious about their machines' WIN.INI and other configuration files are a major source of technical support revenue for them.

Cultural Values

It is said that you can judge a culture by the things it values. If that's true for civilizations, it's also true for computers. Over the years, the graphical interfaces of Windows and Macintosh have gotten more and more similar in appearance, but the thing that sets them apart continues to be the philosophical values that created them.

On the Wintel platform, the buzz is constantly based around the capabilities of the computer. What's the speed of the processor? What sort of cache does it have? How much VRAM has it got? And so on. These are the sort of benchmarks that have always been used to judge Real Computers—with one historic exception.

When the Macintosh was developed, enough talented people got together and decided that the real measure of the computer was how productive the people using it were. More than just a slogan, it shows in every facet of the Macintosh's design—from its graphical user interface to the fact that users could give their files real names ("1996 Annual Budget" instead of license-plate-like, eight-character abbreviations such as "96ANLBUG.DOC"). Even now, with longer filenames ("1996 Annual Budget"—don't forget the ".DOC" or the file won't open) and a more Macintosh-like graphical interface, the Wintel platform continues to be shaped by the technology-centered culture that gave birth to it.

That's why when my cross-platform project is over, the Pentium will still have a place in my office. Its processor works remarkably fast, it's a fascinating machine to program on, and it plays a great game of pinball.

But when I want to get actual work done, I'll be doing it on my Mac.

Till next time,
Doc

Peter Bickford has a new role as a Senior Scientist in Apple's Developer Consulting Group. If you have interface questions you'd like to address to him, write to bickford@apple.com.

The Year 2000

continued from page 15

The World Wide Web site at **http://www.year2000.com** is devoted to the issues related to the year 2000. There is also an Internet mailing list for discussing year 2000 issues. To subscribe, send e-mail to listmanager@ hookup.net with the text "SUBSCRIBE YEAR2000" in the *body* of the message.

You can find a good quick reference to the Gregorian calendar and its evolution at the Web site **http://es.rice.edu/ES/humsoc/Galileo/Things/gregorian_calendar.html**. ♣

Brian Bechtel is an engineer in Apple Developer Support.

FEATURE

Developing World-Ready Multimedia Software

By M. Raymond Jason, International Language Engineering Corporation

Last year Apple Computer, Inc., sold nearly 4 million multimedia-ready computers around the world, and as a multimedia developer, your ticket to this global market is *internationalization*—designing and developing your software product to make it easy to localize for other countries.

Localizing multimedia software for regional markets is a lot like preparing for an around-the-world trip: With a little advance planning, you can be "world-ready" with a minimal amount of hassle. Prepare at the last minute, however, and you're likely to suffer from unexpected delays and expenses.

As a project engineer at International Language Engineering (ILE) in Boulder, Colorado, I often act as an internationalization consultant to developers, offering them suggestions on ways to save localization time and money. This article summarizes the advice that I most often provide my clients, and is targeted especially at those who work with media-rich content.

Multimedia vs. Traditional User Interfaces

Multimedia software is defined by its complex data types, and as you'd expect, the more media elements you add, the harder it is to localize. But since multimedia titles also contain the elements found in traditional software—windows, menus, and dialog boxes—it's sensible to view multimedia as a superset of traditional software.

From a developer's perspective, there's no hard and fast division between multimedia and traditional software. After all, if your program has an icon and contains PICTs, you're dealing with more than one expressive medium.

The commonly accepted guidelines for internationalizing software with a traditional user interface can quickly be summarized as follows:

- Draw on internationalization expertise as early as possible in your development cycle, to optimize your product's structure and content for localization.
- Keep user interface resources separate from underlying functional code.
- Anticipate text expansion.
- Maintain consistency and accuracy of terminology.
- Avoid culturally specific content.
- Use Mac OS resources for date, time, measurement, and currency formatting.
- Use Mac OS resources for text handling (display, sorting, word breaks, and so on).
- Maintain up-to-date, detailed asset information and style specifications.

For detailed information on all of these issues, read the ILE document *Accent on Internationalization: Guidelines for Software Internationalization* (see "Internationalization References" on page 22 for ordering information); "Internationalizing Your Software" in *Apple Directions,* October 1994; "Writing Localizable Applications" in *develop,* Issue 14, page 7; and *Inside Macintosh: Text.*

Internationalization Guidelines—New Media, New Challenges

Now I'll discuss each of these guidelines in the context of new media, because once you add audio, video, or static images to a product, a whole new set of challenges enters the development process. In this section I offer some suggestions for internationalizing new media data types and managing the localization process for multimedia content.

Tip #1: Internationalize Early and Often

The best way to minimize costs and maximize sales in software internationalization is to plan ahead—in fact, long before coding begins.

Efficient localization depends on informed internationalization. So does market success in each of your target regions. Internationalization is the foundation for the whole endeavor, and the end result derives, quite inevitably, from the level of care put into it.

A good localization vendor or in-house expert can offer guidance during product design that sets you in the right direction and helps keep you on track. For instance, a localizer could review your sound design plans and recommend changes in pacing or file format (see the sections "Anticipate Temporal Expansion" on page 6 and "Choose File Formats Carefully" on page 22), or in your data layering strategy (to facilitate replacement of voice over music).

Tip #2: Segregate User Interface Resources

Localizers prefer that developers segregate user interface resources from functional code, because this makes the user interface easy to change without "breaking" the software. This issue is identical to the one faced by all developers moving a product from one version to the next. As a multimedia developer, you're required to segregate a variety of user interface data types, and they each have their own peculiar internationalization requirements. The fundamental principle behind user interface segregation, however, is identical for all media.

For example, audio tracks are typically replaced during localization. This is true not only for recorded voice, but for other sounds as well. New York street ambiance, with American car horns and ambulance sirens, bears little resemblance to the sounds heard in a comparable "scene" set in Paris or Beijing. By simply placing sounds into 'snd ' resources and using the Resource Manager of the Mac OS, you've made the sound easy to replace. (I discuss audio-specific design and internationalization issues in tips #10 and #11, beginning on page 21.)

The Mac OS inherently supports internationalization by organizing files into data and resource forks. If you adhere to the standard Macintosh practice of placing all user interface elements into appropriate resources, you'll satisfy the number one rule of internationalization. Localizers will be able to use standard resource editors, such as Mathemæsthetics' Resorcerer and Apple's ResEdit, and Apple's localization and leveraging tool, AppleGlot, to localize your application's user interface without recompiling. These tools work with any properly designed Macintosh program, whether you develop in C++, Smalltalk, or some other language.

As cross-platform developers know, the Windows 95 world is far messier. There's no universal set of resource descriptions for Windows and no standard scheme for defining custom resources. Localizers must typically

turn to the resource editors included in development packages such as Microsoft Visual C++ or Borland C++ to edit a Windows user interface, depending on which package was used to create a product. And, for a variety of reasons (especially when custom resources are involved), Windows localizers often prefer to receive source files rather than compiled executable (.EXE) and dynamic linked library (.DLL) files. This dramatically increases the size and complexity of the hand-off package you submit to a localizer, and it significantly complicates the engineering aspects of localization, relative to a Mac OS–based project.

Tip #3: Anticipate Text Expansion

The second classic rule of thumb in internationalization is this: Anticipate a 30 to 50 percent character-count increase for text strings, and an expansion of up to 100 percent for individual words. In rare cases, words or short strings can expand as much as four or more times.

Ironically, this rule of internationalization happens to be language-specific: It applies most often to developers working in English, an unusually concise language. User interface designers developing in certain European languages, such as Spanish or German, can often ignore this issue.

When text expands beyond expectations, it affects many things. String buffers just long enough to accommodate English text may overflow when text is translated, crashing your software. Memory requirements and disk-space requirements go up. Buttons, checkboxes, radio buttons, and pop-up menus require resizing.

With new media types, you must consider text expansion in more places. For example, never sandwich text between "previous" and "next" navigational buttons, since localization—and consequent text expansion—could necessitate a code change to relocate the buttons. Instead, place the associated text above or below the buttons. Also, avoid creating graphical buttons overwritten with text, to minimize time-consuming artwork localization.

Here are some other places to watch out for text expansion:

- in movie subtitles and credits, where additional text may overfill the frame
- in animated labels or 3D models that contain text
- in any framed text or text appearing over a map or anatomical image, where you may need to adjust the scaling of the image to display longer words without obscuring the subject matter

Tip #4: Avoid Culturally Specific Content

Begin internationalization before your programmers write a single line of code by designing your product to be culturally neutral. I must admit that this is far easier said than done. In fact, with some multimedia titles it may be impossible, since excluding culturally specific sounds and images can lead to a rather sterile user experience, and this defeats one of the goals of adding media in the first place.

When cultural neutrality isn't an option, you might choose to keep your product within the borders of your home country, or pursue one of two internationalization strategies that I'll describe in a moment. But first I'd like to frame this discussion by reviewing cultural neutrality as it applies to the traditional user interface, and then as it applies to multimedia.

Culturally specific text can include humor, religious or political references, sports analogies, jargon, and slang or colloquial

A Multimedia Developer's Internationalization Checklist

Here are a few multimedia product design tips that can save you time and money when it's time to localize your product for other countries.

__Prepare for a 30 to 50 percent character-count increase for translation of English text strings, and up to a 100 percent expansion of individual words.

__Anticipate a 50 percent overall time duration increase for translations of English voice tracks.

__Position text above or below graphic navigation buttons, rather than between them, so localizers won't have to redo artwork to accommodate text expansion.

__Avoid culturally specific text, such as religious or political references, sports analogies, jargon, and slang or colloquial expressions.

__Use the WorldScript I and II system extensions for defining word breaks, sort order, text display, and date, time, measurement, and currency formatting.

__Access QuickTime's Movie Toolbox and the Gestalt function to present the appropriate language tracks for multilingual movies.

__Provide your localizer with an asset list that defines which data elements contain text, currency symbols, or culturally specific images or audio, and specifies what the localizer must deliver for each such asset.

__Provide your localizer with detailed creative specifications for each user interface asset.

__Make sure you're using art tools that enable your localizer to place, edit, and manipulate double-byte, complex, or bidirectional text, as appropriate for your target markets.

__Use AppleGlot's pretranslation function so your localizer can leverage translations from your previously localized product or Apple's new international glossaries.

__Expand your audio storage requirements to ship audio sampled at 22.05 kHz or higher for languages that are rich in phonemes (meaningful sound units).

__Maintain language-independence for as long as possible in a data type's processing lifetime.

expressions. Avoid these for the smoothest localization experience, or at a minimum, have this type of content reviewed by a native speaker from a target market.

As an example, consider an "OK"-type button that contains the string "Go for it!" Ideally, your localization vendor would catch this colloquialism early on, during your project's pretranslation and glossary preparation stage. At this point your designers and in-country reviewers would decide whether to maintain the "feel" of the phrase for each target locale or to simplify matters by replacing it with the less colorful, more functional standard term. Without this kind of early consideration, the problem wouldn't be noticed until the translation process, or, worse yet, until in-country review of the localized software. In this sort of situation, you would have to pay your localizer to independently track and resolve each such issue for each language version.

Culturally specific video and audio content is harder to avoid. Your choices range from keeping your content culturally rich and paying for new production from scratch for each language, to carefully screening out cultural sounds and images to allow reuse of most of your original production work. Since each case is unique, consult an experienced localization vendor for guidance in this matter.

With the benefits of cultural neutrality in mind, here are two internationalization options for products that must contain some (or a great deal of) culturally specific content.

First, you can position your product differently by redefining the role of the content. For instance, you could take a vocabulary-building game for American children, and market it as an English-teaching application for Chinese children.

Second, you can partition your content—from the start—into culturally neutral and culturally specific components, then replace the culturally specific content appropriately and separately for each target market. The expense incurred by this option comes, of course, from developing an additional subset of original content for each market. The savings come from two fronts: You avoid having to pay a localizer to determine which pieces of content require change, and you can better estimate the scope of the localization work in advance. To pursue either option most effectively, engage localization expertise early in your planning stage.

Tip #5: Leverage Mac OS Resources

The Mac OS, by way of its WorldScript I and WorldScript II extensions, supplies you with a set of internationalization resources that simplifies the display, formatting, sorting, and editing of user interface content. By using these built-in resources through the Script Manager, you can internationalize most text, time and date, and currency functions for most of the world's languages. (See *Inside Macintosh: Text* for details.) The forthcoming release of Mac OS 8, with its built-in Unicode (ISO-standard, universal character encoding) support, will make it even easier to internationalize the text portion of your user interface.

Although Windows 95 also provides international resources, the implementation is significantly less elegant than that of Mac OS 7.5.3. For example, with Windows 95 it's impossible to compile for all target languages on a single platform. To compile Japanese, Chinese, or Korean versions of your product, a localizer must run the IDE (integrated development environment) on the target local operating system. If you want to sell a Windows product into each of these growing Eastern markets, and into Europe as well, a localizer may need to set up, run, and maintain as many as four OS and IDE installations for your project. To localize a Macintosh product into any language, a localizer just needs the standard U.S. English operating system version and the appropriate language kits.

QuickTime movies (time-based multitrack sequences containing any combination of animation, video, MIDI information, text, and sound) can include specific tracks for each of your target languages. Use QuickTime's Movie Toolbox to present the appropriate language based on language and region codes, which you can obtain through the Gestalt function (see *Inside Macintosh: QuickTime, Inside Macintosh: Text,* and *Inside Macintosh: Operating System Utilities*).

Tip #6: Document Your Assets

When you hand off your original software to a localization vendor, you are in a sense enlisting a parallel development team. Localizers rarely need to alter your code, especially if you've followed good internationalization practice. But a localization team must learn your software's resources and interface inside and out, sometimes in greater depth than any one person on your internal development staff. After all, a localization team's charge is to produce as many as 20 or more flawless versions of your product.

Remember that your localization team is at a great disadvantage relative to your in-house developers. They haven't participated in the design of your product, or lived with your code for months, or learned through internal memos and hallway discussions about the "tricks" and "gotchas" lurking in your custom resources. They'll need a huge amount of background information to do their work. Nonetheless, your schedule will probably demand that this team ramp up to production speed as spontaneously as the starship Enterprise accelerates when Captain Picard from the Star Trek television show intones "Engage, Mr. Data."

It's not dilithium crystals that make such unnatural acceleration possible, but rather a detailed asset list. (If you have some spare dilithium, however, please e-mail me some.) Such a list should include information on every resource you want localized, and it should be part of a "localization guidelines" package that accompanies every localization hand-off. If you've never created such a list, request instructions and a sample list from your localization vendor.

When your product includes a variety of data types, the asset list should be correspondingly larger and more complex. For example, if your software contains 3,000 bitmaps, your asset list should define which ones contain text, currency symbols, or culturally specific images, and it should include instructions to the localizer for each item. If your software uses 500 sound clips, your asset list should specify which ones contain voice or culturally specific music or sound effects, and, again, what you want done with each clip. If synchronization is an issue for any time-based media, your asset list should include time-code in and out cues to streamline production work.

Tip #7: Pick Art Tools Carefully

Multimedia localization often involves modifying artwork. This ranges from simple tasks, such as replacing a layer of bitmapped text, to complex ones, such as replacing animated text in custom buttons or in a QuickTime movie.

For the smoothest localization, ask your internationalization expert a few key questions on the drawing, painting, animation, and authoring tools you plan to use. Can your localizer use the same tool to place,

edit, and manipulate double-byte text? (Double-byte text is used for large writing systems—such as Chinese and Korean—that require 2 bytes per character, rather than the single byte per character that suffices for most of the world's writing systems.) What about support for complex or bidirectional writing systems, such as Arabic, Hebrew, and Thai? (Bidirectional systems are those in which text can be inserted in left-to-right and right-to-left formats in a single document.) Depending on your target markets, these questions may be critical. Few art or authoring tools sold in the United States support non-Roman scripts, and some aren't even available in double-byte or bidirectional versions. Since this support situation changes rapidly, I recommend that you consult your localization vendor before committing your artwork development or multimedia authoring to one tool or another.

Tip #8: Check Language Support of New Technologies

The availability of a new technology in the U.S. version of the Mac OS doesn't guarantee availability in all languages. For example, Apple Guide employs an internal search engine that is, by necessity, language-specific for stemming. (Stemming is the reduction of word forms to the associated root word; for example, Apple Guide stems *edits* and *editing* to the common root *edit* before searching.) Until recently, Apple Guide searching was unavailable for most languages.

While Apple Guide now supports 21 languages, third-party help systems may be more limited. A localization expert can help you choose an online-help authoring environment that provides the double-byte or complex language support that you need.

Be aware that Apple's PlainTalk text-to-speech technology is still only offered for U.S. English, Chinese, and Mexican Spanish. PlainTalk speech recognition is only available for English. You can't, at present anyway, include PlainTalk's features in your software for other languages.

Tip #9: Take Advantage of Pretranslation

AppleGlot's pretranslation function can save you a load of cash by helping your localizer leverage translations from your previously localized product, or by leveraging from Apple's new international glossaries. To make it easy for your localizer to pretranslate, carefully archive parallel (original and localized) versions of your product and include them in your localization hand-off package.

Apple recently surpassed Microsoft in the pretranslation arena, by the way. The Windows 95 glossaries available to developers and localizers are raw OS dumps, containing thousands of duplicates and unusable long strings. Apple's glossaries, in contrast, are—well, plug-and-play. You can find these new glossaries on Apple's Worldwide Developers Conference (WWDC) CD.

Tip #10: Anticipate Temporal Expansion

Because text tends to get longer when translated (at least if the original software is in English), voice segments tend to take longer. The time ratio is roughly the same as that for textual expansion—anticipate a 30 to 50 percent overall increase in duration.

Internationalizing for temporal expansion means three things. First, you should expect to fill up more of your CD-ROM. One second of low-quality (8-bit, 11.025-kHz, monophonic) sound consumes about 11 KB, while one second of CD-quality single-channel audio consumes eight times that amount. It doesn't take too many additional seconds at either quality level to add up to significant disc real estate. If your original CD already contains 450 MB of data, you might be in trouble.

Second, ensure that events that depend on the completion of a sound clip always

A Creative Specification Checklist for Multimedia Voice Tracks

Parameter	Suggested options
Speed of voice delivery	__Slow enough to remain intelligible when reproduced over a low-quality computer loudspeaker, possibly at expense of interest __Fast enough to remain interesting, possibly at expense of intelligibility
Emotional content	__Always friendly __Appropriate to role implied
Emotional level	__Similar to U.S. version __More emotional than U.S. version __Less emotional than U.S. version
Humor	__Appropriate to content __Always funny __Type of humor: playful, goofy, hilarious, or tongue-in-cheek
Perceived age of adults	__Appropriate to implied role __Young __Middle aged __Elderly
Perceived age of children	__3 to 5 years __6 to 9 years __10 to 12 years __13 to 15 years __16 to 18 years
Gender	__Appropriate to implied role __Female __Male

wait for completion, and not just until a certain amount of time has elapsed. And design the user experience to accommodate this type of delay in a way that isn't off-putting or confusing.

Third, build strategic pauses into your product as temporal buffers for additional speech time. This guideline is especially important for narration associated with movies or screen-based training. Strategic pauses let you avoid the cost of reproducing or reediting video or screen activity to accommodate the natural pace and duration of a localized voice track.

Tip #11: Choose File Formats Carefully

When you intend to localize, your archival and commercial file formats must satisfy not only your developer's and designer's requirements, but those of your localizer and your expanded target market as well. Factors to consider are processing and cross-platform issues, international standards, end-user expectations, and future trends.

While most audio formats translate almost effortlessly across platforms and across national boundaries, video is more fussy, with the coexistent PAL, SECAM, and NTSC broadcast "standards" along with the QuickTime video standard. What's more, with DVD and HDTV looming over the horizon, and with Microsoft actively promoting ActiveMovie as a QuickTime replacement, predicting what will work down the road gets a bit risky. Enlist a localization vendor at your planning stage to help work out a production strategy that makes sense for each of your target markets and for the expected lifetime of your product.

Target-market audio quality expectations may be tougher than those you're used to, especially if you're a developer based in the United States. Many European languages, for example, contain a richer variety of phonemes (meaningful sound units) than does U.S. English, including a wider variety of sibilant sounds. It's extremely difficult, if not impossible, to satisfy European ears with an 11.025-kHz voice track. Unnatural sibilance comes across as a faulty accent or a speech impediment. You'll have much better success if you can afford to double your storage requirements and ship audio sampled at 22.05 kHz or higher.

Don't fail to clarify right up front what you expect regarding deliverables for your multimedia data. If you work with 20-bit audio internally to future-proof your efforts, specify that you want localized 20-bit files returned along with the reduced-size, commercial-quality audio. You won't want to throw away the quality you paid for when new, higher density (holographic-crystal?) storage becomes standard in a couple of years.

Tip #12: Strive for Language Independence

Even simple localization requires more than just language translation: There also needs to be one or more competent software engineers on the localization team. And localization of the more esoteric data types can require quite intensive engineering. During multimedia development, it makes good sense to think of the engineering process itself as something you can internationaliz.

The goal is straightforward: to minimize the work needed to produce the desired deliverables. The better you understand the localization process, the more effectively you can design toward this goal. The essential rule of internationalization for your multimedia data engineering process is to maintain language independence for as long as possible in a data type's processing lifetime.

Consider the case of producing and then localizing an audio clip consisting of voice and sound effects. It takes your engineers perhaps eight steps to convert original digital audio recordings to the final sound resource format shipped on your CD-ROM discs. The steps might include compression, equalization, addition of effects, mixing, dithering, requantizing, resampling, and, finally, file-format conversion.

Some of these steps may add value and enhance quality; some "trap" the language-specific content, irreversibly combining it with other audio elements; and some compensate for limitations in the delivery medium. (Resampling and requantization fall into this category.) Still others (for example, compression and equalization) may assume any of these roles. To internationalize, you should first perform all processing steps that add value. Then "trap" the language content as your sound design calls for it, and, finally, perform the steps required to compensate for the delivery medium.

This strategy enables you to provide your localization vendor with as many "prefabricated" parts as possible, and allows for a minimum of engineering and creative backtracking to replace the language-specific content.

Tip #13: Maintain Up-to-Date Creative Specifications

To get what you want, you must communicate—in some detail—what you expect your localization vendor to deliver. And with multimedia software, it isn't enough to supply engineering specifications, an asset list, and an approved glossary. You should also hand off a set of creative specifications.

For an example of the type of creative information a localizer needs to efficiently execute a multimedia localization project, see the checklist on page 21. The relevant creative specifications will, of course, depend on the nature of your product. For example, if you don't employ spoken language, you need not specify the real or apparent age of the actors.

A Final Word on Internationalization

Internationalization is akin to saving up for your child's college expenses. A small up-front investment can be the best way to avoid financial "discomfort" later on. But this analogy isn't quite fair, since the extra documentation, planning, and structure that you invest during internationalization tends to benefit your own efforts as they're happening. The cleaner the coding, the less debugging. The more consistent the design, the shorter the quality assurance phase. Multimedia data types simply amplify the familiar problems and benefits of software development.

Internationalization References

• *Accent on Internationalization: Guidelines for Software Internationalization,* available from International Language Engineering Corporation, Boulder, CO 80301; phone: 303-546-8260, e-mail: localize@ile.com, Web site: http://www.ile.com.

• "Internationalizing Your Software," *Apple Directions,* October 1994, page 18, presents an overview of script-system internationalization issues.

• "Writing Localizable Applications," *develop,* Issue 14, page 7, and *Inside Macintosh: Text,* provide Mac OS–specific guidelines for international resources.

The care you put into internationalization will also tend to streamline your in-country updates by giving new team members valuable insight into the minds of the people who created the previous version. For expensive data, such as mixed audio or lip-synched video, the design strategies I've outlined here will ensure maximum reusability.

Localizers deal with this article's issues every day; take advantage of their guidance before you head down a path that will unnecessarily inflate your costs or limit your options. With good planning, you can start to think of the entire world as your target market. ♣

The author of this article, M. Raymond (Murray) Jason (mrj@ile.com), is a project engineer at the headquarters of International Language Engineering Corporation (ILE) in Boulder, Colorado. Jason specializes in the technical and creative aspects of Macintosh and Windows product localization, including localization of multimedia titles. Over the last 12 years, his company has localized a wide variety of software products, online help systems, and documentation into as many as 23 different languages.

Issue 27

continued from page 14

facilitate OpenDoc part editor unloading. And some old favorites are back: Dave Johnson writes about learning pool vs. learning programming, and KON and BAL present a Puzzle Page conundrum that you might actually be able to solve.

So please check us out on this month's Developer CD, on the Web at **http://dev.info.apple.com/develop.html**, or in print if you've subscribed to *develop* through the *Apple Developer Catalog.* As always, we'd appreciate receiving your no-holds-barred feedback, on this issue specifically or on *develop* in general, at develop@apple.com.

Caroline Rose, Editor, develop

CD HIGHLIGHTS

Reference Library Edition

continued from page 14

MacODBC 2.1.2 GM

MacODBC is the standard way for developers to write Mac OS–based applications that communicate to databases using the Open DataBase Connectivity (ODBC) standard. This folder contains the software development kit for MacODBC version 2.1.2 GM. Features include

- cross-platform remote database access
- a wide range of third-party drivers
- coverage of all major relational databases
- support for many Mac OS databases

MacsBug 6.5.3

This is the latest version of MacsBug, Apple's object-level debugger. See the document MacsBug 6.5.3 Read Me for details about this release.

MenuScripter 4.0

The MenuScripter sample code demonstrates advanced features of the Open Scripting Architecture (OSA). The OSA allows you to alter MenuScripter's behavior by attaching scripts to objects such as documents. You can create the scripts in the Script Editor, in MenuScripter itself, or in any other script-editing application.

MenuScripter 4.0 implements many of the techniques described in Paul Smith's *develop* articles "Programming for Flexibility: The Open Scripting Architecture" and "Implementing Inheritance in Scripts." Anyone who intends to use the OSA should read these articles.

Here are the OSA features MenuScripter demonstrates:

- attaching scripts to objects
- installing a predispatch Apple event handler to give an object's script the first opportunity to handle any events targeted at the object
- directly compiling and executing scripts
- decompiling existing scripts for editing
- getting and setting properties in scripts
- retrieving information on errors that occur in a script
- loading and storing scripts

MoreFinderEvents

The Apple events API is icky and you probably don't relish the idea of developing an intimate relationship with it. And you don't feel your application should require the scriptable Finder, which first shipped in Macintosh System 7.5. Yet you still want to make the Finder do backflips and cartwheels. Enter MoreFinderEvents. This package provides you with a painless API for sending some of the more simple forms of the Finder events, which have been with us since System 7.0.0. With this API you can copy a file, open a control panel, empty the trash, or do any of several other things usually done by the Finder at the user's request.

OTPingSample

This sample is a quick demonstration of how to implement the ping function on top of the Open Transport native APIs.

OTTraceRouteSample

This sample is a quick demonstration of how to implement the traceroute command on top of the Open Transport native APIs. The sample demonstrates how to set the IP TTL option and how to use a "rawip" endpoint to receive and process ICMP packets.

Scrap Color 1.0

Scrap Color is a Clipboard format and supporting code library to help communicate color information between applications by means of the Clipboard. It was released in June 1996 by Mark Womack and Patrick Bores. (The Code library was written by Mark Womack.) See the file Scrap Color Documentation for details.

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

ZapTCP 1.2.1

ZapTCP provides a safety net for MacTCP developers. It automatically cleans up TCP streams that are left dangling when your application quits unexpectedly. This prevents a lot of restarting during MacTCP development. Version 1.2.1 renders the extension benign under Open Transport.

Alex Dosher
Developer CD Leader
and Online Content Librarian

Business

Feature: Five successful Mac OS developers talk about how they turned a few good product ideas into highly profitable businesses. They also discuss platform strategies, financing, product timing, and exit strategies.

How Five Mac OS Developers Made a Million

Based on Guy Kawasaki's WWDC Developer Panel, "How to Make a Million"

Some developers program on the Macintosh for love. Some developers do it for money. And along the way, some of them become millionaires, often quite accidentally. In this article you'll read the stories of five Mac OS "alchemists" who turned a few good ideas into gold.

Though all the developers in this article made their millions in different ways, there are several success factors common to each. One characteristic they shared was the ability to put initial failures behind them, then move on to the next big idea. None of them started their businesses with large bankrolls: They all bootstrapped their way through with the help of savings, credit cards, and friends. All of them were willing to work very hard. And finally, call it luck or call it timing, they all had the right idea at the right time.

These "accidental millionaires" also shared their thoughts on some issues relevant to many developers today, such as

- Macintosh-first versus cross-platform development
- which type of financing works best: venture capital or bootstrapping
- startup "exit strategies" and when to "cash out" of a software business
- whether now is a good time to start a new software company
- whether you'll feel guilty after making a million dollars (just kidding)

Perhaps the most valuable thing you'll take away from these stories, however, is inspiration: Each one is an affirmation that you can do what you love—Mac OS software development—and be handsomely rewarded for it.

• • •

Bob Hearn on the Making of ClarisWorks

"We knew we couldn't develop the product that we had in mind inside a big company. In 1989, Scott Holdoway and I were working at Claris because our company, StyleWare, had been bought by them. We'd just finished AppleWorks® GS, and we wanted to do a version of this integrated package for the Macintosh. Because the idea wasn't very popular at Claris, we decided to take some recent bonus money and develop the product ourselves. For the first few months we worked out of a dining room, and our startup costs were basically two Macintosh IIx computers and enough money to live on for a year. Everybody told us we were crazy to compete with Microsoft Works, but we figured, what the hell, we'd give it a try.

"When we left Claris, our original goal was to sell the product back to Claris. All our friends were at Claris, and we sort of liked the place. So we worked on it for about nine months, tried to sell it back to Claris, and they didn't want it. We then had to decide between selling it to somebody else or looking for venture capital, but we couldn't get in the door anywhere. After all, we were just a couple of naive kids who knew how to program, but didn't know anything about business. Eventually, we hooked up with Guy Kawasaki, who saw our product and said, 'Move into my office, I'll move out.' (Too bad he can't do that for all of you—he doesn't have that many offices.)

"For a few months we worked in Guy's office, until we got some competitive bids. Eventually we sold the product to Claris for some cash and royalties. Since then, we've developed two new versions for Claris under separate contracts.

"When we first left Claris, everybody told us, 'You'll never make it, this is not the same market as it used to be. It's too late, because the dominant players are established, and you can't compete with Microsoft Works.' But here we are today with a successful product. Overall, I think the ClarisWorks deal worked out the best for all concerned, and we're hoping we can do it again."

Terry Morse on the Making of DiskDoubler

"Unemployment spurred Lloyd Chambers and me to start our own company. We were both at Mansfield Systems. I had been there for a month, working for Lloyd as programmer, when our venture capitalists told us that they weren't going to fund us anymore. We were out of a job. We both thought about the prospect of writing a résumé and putting on our interview ties, but it didn't sound like a whole lot of fun.

"At that point, we decided to write some software of our own, just to see how far we could go. A month later we formed a partnership, and two months later we shipped our first product. It was a product called *Partner,* which was basically a poor man's OpenDoc for System 6. It was very primitive, and it didn't go anywhere. Then, over a weekend, Lloyd wrote DiskDoubler, and that product really seemed to catch on.

"From there we built a company, and by 1992, we had about $7 million in sales. Once you get to that sales level, other companies begin to look at you as a buy-out target. And such was the case with our company: In about two years and six months, we went from the ranks of the unemployed to entrepreneurs who sold out to Fifth Generation Systems, which is now part of Symantec.

"It never really occurred to us to ask a venture capitalist for money. Lloyd used his stack of credit cards, so we were as well funded as we needed to be. After about six months, we showed DiskDoubler to Symantec, and they told us to get lost. So Guy offered us $25,000, and we said, 'Wow, a year's worth of living expenses.' Then we took another $25,000 from

some other people and used it for advertising. We became profitable in October 1990, when we started selling DiskDoubler nationwide.

"Venture capitalists came to us in the subsequent months, saying, 'Want a million dollars? Want to build a Windows version?' But we told them no thanks—we're having fun doing this *and* we're making money.

"Business was pretty strong in early 1992 when our outside investors pulled me and Lloyd aside, saying, 'You know, I think it's time to sell out.' We told them no way. We're going to reach $4 million in sales this year. They chuckled and said something like, 'baloney,' but we reached two million in sales in about two-and-a-half months. That also caught the attention of other investors.

"People from Fifth Generation met with us to discuss buying our company, but after the initial meeting, we didn't hear from them for about two months. Finally they called us back, invited us to dinner, then offered us a cash deal. We left the dinner, talked amongst ourselves, then decided to take it.

"To add to that, the business of business is making money, and when you're able to do that, you should run with the money. In hindsight, I had a personal stake in the company, but we had investors, and they needed to be paid. It's cold and brutal, but that's business.

"Lloyd and I had signed a two-year contract to work for Fifth Generation, so after the sale we continued to do development work and I would regularly harangue them over their marketing decisions. Fifth Generation was ultimately bought out by Symantec, and Lloyd's still an employee there. Now I'm out on my own again.

"For those of you thinking about going out on your own, consider this: All along the way people told us we couldn't start a company without capital. Or that we couldn't publish a product on our own. But we did it. And, yes, things have changed today: There are twice as many Macintosh computers as when we started. There are bigger software companies now, but once they get gobbled up by each other, it will be harder for them to develop new technologies in-house. These large businesses are a ready market for innovative products coming from independent developers."

Lloyd Chambers With More on DiskDoubler

"When my employer's funding ran out, I decided I'd rather start my own company than start a job search. One lesson that I took away from this experience is this: If you're forced into what looks like a situation you don't want to be in, sometimes there's hidden opportunity. If those venture capitalists hadn't stopped the flow of money into our company, I might still be there, and it wouldn't have been possible to build the business that we did. So I feel fortunate that things happened in that order.

"In about two years and six months, we went from the ranks of the unemployed to entrepreneurs who sold out to Fifth Generation Systems, which is now part of Symantec."

"We used credit cards to fund our company for a period of time, but I got pretty nervous when the cards began to reach their limits. It was about that time that we managed to procure some funding and actually draw a small salary. The risk was bankruptcy, but at that time I was a student with only a 1978 Toyota with 120,000 miles on it and a bicycle. I didn't have much to worry about.

"Starting a company with too much money doesn't seem to work as well. (You need just enough money to buy the latest, most powerful Mac. I had to schlep my Macintosh IIfx computers between home and work for six months until I could afford another computer.) You have to stay lean. It helps you keep your focus. If you need to hire somebody, you make sure you hire a good person. You can't afford to hire some bozo who's going to take three months to learn the ropes. (Our idea of a big expense was when I flew to Oregon for three user group meetings. My total expenses for the week, not including airfare, were about U.S. $150.)

"Sometimes when companies have too much money, they think that hiring twice as many programmers will get the job done in half the time—it really doesn't work that way. With every hire you get more personnel management and large-company problems. In a startup you can't afford to lose your focus.

"I think this philosophy kept us from developing a Windows version of our product. Doing the 'cross-platform thing' introduces a tremendous amount of variability in your planning process. You need a whole different set of skills, you need Windows people, and then you suddenly have several people to manage. At the time, I really wanted to stay focused and push as fast and as hard as we could on the Mac version. If we'd done a Windows version, we would've been constantly forced to assess which features to compromise, since you can do some really cool things on the Macintosh that you can't do on Windows.

"One thing I recall about the process of selling our business was that it really required a change within me. We were having a great time doing our own thing, but when someone offered money for our company, it took about six weeks to decide that it was the right thing to do. It was a hard decision, because I had coded a lot of blood and sweat into the product. I think anybody who has a keen an interest in what they're producing will probably go through the same sort of thing. Just remember, you have to think about what you're really doing this for: Is this what you want to do for the next ten years? If the answer is no, then cashing out is the right thing to do. In retrospect I'm very happy we made the decision to sell.

"Once you sell a product, you have to accept that somebody has paid you money for it, and now it's theirs to screw up. This is usually tough to take. Fortunately, we were able to maintain some degree of technical control. Again, you have to take the deal as a whole and realize that you've 'sold the car and now it's somebody else's to drive.' Get your money up front and make sure it's enough to make you happy to relinquish the enjoyment that you've gotten from it.

"In looking at the software channel today, you'll often read about how hard it is for Macintosh software to get on the shelf. It was pretty difficult for us to get shelf space four years ago, and it's probably much more difficult now. On the other hand, things like the Internet may provide new opportunities for Macintosh software, enabling small startups to make a good living with an innovative little product. If you get lucky and have the right vision,

maybe you'll create a great product that some public company will pay a ridiculous amount of money for."

Richard Wolpert on the Making of TouchBase and DateBook

"I had no mortgage, no wife, and no liability; and my 'old' forty-something friends told me it was time to start a company. I had been working at Apple since 1985, when I was offered a job promotion. It would have increased my annual salary by about $20,000, and I thought, 'Wow, I could buy a car and maybe a house.' But after thinking about how those financial obligations would trap me, I decided to leave Apple. I started consulting, doing the same type of work I was doing for Apple at home, and making about twice the money. It was a *good thing.*

"As a consultant, clients started giving me more and more work, and I started hiring people. Within a couple of years I had a staff of about ten people. For about three years, we were doing some pretty interesting database consulting, but I really wanted to make a lot of money, and I realized that there was no easy way to sell a service business. So in the late 1980s, we transitioned the company to retail products. In 1989 we shipped TouchBase, in 1991 we came out with DateBook, and in 1992 we sold both of these products as a bundle. In 1993 we came out with a product that we called *GUM,* Guy's Utilities for the Macintosh. (Before Symantec killed this product, they called it *Norton Essentials for PowerBook*.)

"I started my business with $30,000 in savings and that took me through the first four or so years. At one point we were making a really big retail push. CompUSA really started coming on strong, and things like end caps started to get a lot more expensive. We needed an additional $200,000 to $250,000 to really make a difference. What I did, which actually worked quite well, was to open funding to people within the company. We had about 30 employees at the time, and I said, 'Here's where we are, here's where we're going, and here's how much money I'd like to raise. If anybody is interested in investing, let me know.' Within two days we had about five people in the company whose parents or friends committed about $200,000. About a week later a friend told me I was crazy because I was going to have employees' parents calling me up saying, 'What the hell are you doing?' Luckily, it all turned out OK.

"We actually created TouchBase for Windows, which most people probably haven't seen, because it didn't see much light of day. I realized in about 1992 that there was a 'Windows issue.' When we were approached by investors, they always wanted to know about our 'Windows plans,' and having one helped us sell the company.

> **"Once you sell a product, you have to accept that somebody has paid you money for it, and now it's theirs to screw up."**

"For example, there's a company that a couple of us were involved with called *Bit Jugglers.* They created the Macintosh desktop screen saver called *UnderWare.* We actually discussed this issue at all of our board meetings: Should we come out with a Windows version? We heard the gospel of 'delusion of focus.' We heard the gospel of 'It would help me sell the company.' And even though I think the Macintosh is a much better platform than Windows, from a business perspective, you have to look where your customer base is going to be and how that will affect your business. In the end, we convinced Bit Jugglers to create a screen saver engine for Windows, and subsequently, they ended up being bought by Compaq. So that's not necessarily a prescription for the way to do things, but when speaking to Apple developers, I think it's important to show that there are lots of different sides to this issue.

"In June of 1993, we sold our company to Aldus, which Adobe bought four months later. I spent about a year as an Aldus/Adobe fellow, and since then, I've been doing independent projects. I didn't have any problems selling the company, because my end game has always been to sell what I created. I loved what we were doing, and I loved the products, but I had no desire to run a company for 10 or 15 years or to take a company public. I've met a lot of people who almost feel bad saying that they want to make a lot of money doing development. Usually one of the main reasons people start a business is financial gain, and I don't think that's anything to be ashamed of.

"In terms of cashing out, I had a 'mini cash-out,' then a real cash-out. The mini cash-out came with *Guy's Utilities for the Macintosh,* PowerBook edition. On July 1 we saw that a PowerBook was coming out in the September/October time frame, and that it had some operating system deficiencies that we could capitalize on (such as a lack of battery saving utilities). We had the idea to make a product that would be the ultimate utilities product for the PowerBook. We thought that we could probably get Symantec to buy this product, so we called the product *GUM,* as a take-off on Symantec's Norton's Utilities for the Macintosh (NUM). We even photographed Guy with arms crossed, in the same pose as the Peter Norton box photo. We did everything we could to make Symantec like the product, and sure enough, about two weeks before shipping, they called us up and said, 'You can't ship this thing. We want to buy it.' That deal was completed in about a week. Symantec bought all of the rights to the product with the money up front, and there were no royalties.

"The more 'serious cash' came when Aldus called us out of the blue, saying that they wanted to be in the Personal Information Manager (PIM) business. They met with us over lunch at Macworld in January; then we didn't hear from them for two or three months. Finally, they called us in April and said that they wanted to buy us, but it would take a couple of weeks to get an offer together.

"When we were bought in June 1993, I signed a two-year deal with Aldus to integrate our products into their organization. About two months after we were acquired, the Adobe/Aldus negotiation started and it was clear that everything was up in the air.

"Luckily, I had a really wonderful contract worked up by Steve Bochner at the Wilson, Sonsini, Goodrich, & Rosati law firm. During negotiations, Steve Bochner gave me an important piece of advice. He told me, 'You're going to get some money or stock up front, you're going to get some guarantees, then you're going to hear about all this upside potential. You have to accept the deal with the assumption that you'll never see another dollar in the future. If you're not happy with the

deal under those conditions, then you shouldn't take it.'

"One thing that I'd do differently today is that I wouldn't try to act as our own publisher. With our products, we set up our own distribution to the channel and bought retail shelf end caps. Unless you have a lot of money and thick skin, I'd stay away from that end of things. Pick a good product niche, then team up with a distribution partner.

"While growing a business, I also learned a really good personal lesson. In the beginning I thought I wanted to make a lot of money and then not work. My father still works. He's 60-odd years old and he still kills himself 14 hours a day. So after I sold the company, I sat on the bench for a year. The first four months were great, because I'd worked really hard. During the second four months, I got a little bit anxious. After the last four months my wife said, 'Uh, honey, why don't you go get a job.' Over that next year I started getting involved with a lot of projects, and I found that I was still working 14 hours a day when I picked something that I liked. I wasn't working because of the money. Ultimately you just need to pursue something that you really like.

"A friend of mine once told me that the three best things about making money are that you don't have to worry about your mortgage, you can always buy fresh-squeezed orange juice, and you can valet-park anywhere you want."

Robert Seidl on the Making of PageMill

"You actually don't need that much money to start a software company: You just need a couple of computers, some long work days, and a lot of luck. The one thing we wanted to do before we got too far into the development of PageMill was to make sure that we had real customers. We began programming the first prototype in Windows, but when we went out to talk with our target customers—marketing communications people—we found that they were all using Macs, even when all the other employees at their site were using PCs or Sun workstations. Based on that research, we decided to develop our product on the Macintosh first.

"We started coding in March, then in July we started showing the first versions to outside people. During the product viewings, we actually received a lot of customer feedback that told us we should split the product into two pieces with different price points.

"We experimented with venture capital early on, not because we really needed it, but because we wanted to do a test run with this type of investment firm. (We decided to use real beta software before we went out to look for more money.) First, we had to convince them that our business was well thought out because (a) they didn't believe that an inexpensive product would make a good startup business, and (b) they didn't believe that the Macintosh was the right platform to start on. It took a while for them to see that it was a good thing.

"The three best things about making money are that you don't have to worry about your mortgage, you can always buy fresh-squeezed orange juice, and you can valet-park anywhere you want."

"Guy saw our product the week before August Macworld 1995, then he went on a promotional rampage for us. It was great. Our company relied very heavily on Apple for comarketing, and this is something I'm trying to help other developers with now that I'm at Adobe.

"In September we started beta testing, which was actually a lot of fun. As soon as the test began, our phones rang off the hook, with people from all over the world giving us Visa card numbers and checks. We kept all the money in a shoe box, since the bank wouldn't give us one of those 'ching-ching' credit-card things. And at one point we found ourselves in our small office wiring our own phone lines, because every time the telephone guy would come to add more lines, the phones of the business next door would go down, and they'd blame us.

"In October we met with a venture capitalist firm, and it was hard work to convince them that creating a Mac product first was the right thing to do. We told them that we had done our homework, and that we had talked to people in the target market, and eventually they came around to our thinking. We were almost ready to sign the deal with them when some other companies phoned up in the space of one week, saying that they wanted to buy our company. Adobe was one of them, and on the phone they said, 'OK, we'd like to do the deal with you. Is next Monday good for you?' So ten days after we had first talked and run a demo, we had a signed deal in hand.

"By the time we were in negotiations with Adobe and others, people were calling us up from the east coast, saying, 'We heard that you're an Internet startup, and that you might need five million bucks from us.' Those valuations seemed crazy to us.

"In our startup phase, we were running on close to zero capital, in spite of having mortgages and wives (and some of them are still with us). We had just enough money to print 3,000 boxes of PageMill at ten bucks each. We thought that was a good enough run for the first quarter, because, hey, that was what was in the business plan. But once we were bought by Adobe, and they began shipping PageMill, they sold 3,000 copies *the first day,* then 30,000 *the first month.* Now we're up to 100,000 or so. We couldn't have done these kinds of sales alone, so it was the right combination of assets at the right time.

"Speaking with the benefit of hindsight, there were two reasons why it was good that we developed on the Macintosh platform first. Number one, Apple was instrumental in helping us become a salable company: They bundled PageMill with their Internet Server Solution, and they helped us with comarketing through print ads and mentions at Apple keynote speeches.

"Number two, the Macintosh market is easier for a startup, because you don't have as much competition. If you look at the Windows market, it's like a Moroccan street market. Everywhere you look there are 15 sellers saying, 'Oh, come here friend. Come, come, come. . . .' The Macintosh market is a bit like a Gothic cathedral full of believers. If Apple believes in your product, they'll put you on an altar. When you shout, you'll hear your

echoes. We're creating a Windows version now, but that's mainly because our market has expanded into corporate intranet authoring, where Windows dominates.

"We started out thinking that we'd go the traditional 'get venture capital, then grow the business' route, until we realized how hard it was going to be to grow the business as fast as it needed to grow. We realized this when Egghead and Ingram distributors called us to say, 'We've got customers asking for this PageMill thing from Seneca, could you please ship it to us?' We'd then have to say, 'Well, we're *not* actually shipping.' With just four of us in the company, we saw how crazy it would get if we had to start selling, setting up distributorships, and doing contract negotiations.

"Because we teamed up with Adobe, we're doing things that we couldn't have done as a separate company. The moment we joined, we were assigned a technical support team of six people in Seattle, which more than doubled the size of our company. They also provided us with people to do the technical documentation, sales, and channel negotiating. (If you've ever put stuff in a catalog, you know how expensive it is to negotiate and get products included.) Now we're able to do Japanese, German, and French versions. In a big company, of course, you do lose some amount of control. You have to find ten other people to agree with you before you can do anything. I guess when that gets too painful, you just go out and start something else.

"You actually don't need that much money to start a software company: You just need a couple of computers, some long work days, and a lot of luck."

"I think Silicon Valley is on a ten-year innovation clock. In 1965 integrated circuits were hot. In 1975 Apple computers and the microcomputer came along. In 1984 and 1985 Adobe PostScript™, Apple, and Aldus created the desktop publishing revolution. And now, ten years later, Internet is the next big thing. This is a very exciting period. Small companies are growing like mushrooms, and even though some of them are getting eaten up by big companies, there are a lot of opportunities for small companies to challenge orthodox wisdom and create new types of products. I think Macintosh developers should take these chances." ♣

The video transcripts on which this article is based were edited by Kris Newby (newby.k@applelink.apple.com), the business editor of Apple Directions *and a freelance writer based in Palo Alto, California. To order the videotape of this developer panel, contact GT Recording in Seattle, Washington at 206-783-6911, and ask for WWDC 96 Tape 701, "How to Make a Million."*

Listings

IT SHIPPED!

The following 58 products were added to the It Shipped! database between June 15 and July 15, 1996, bringing the total number of products entered into the database to more than 620 since Apple reinstated the program in late 1995. Products developed first for the Mac OS platform are designated with the words "Mac first" on the far right; products available only for Mac OS systems receive the designation "Mac only."

The It Shipped! database is an increasingly important tool used by Apple employees for preparing advertising, collateral, and white papers and for helping customers find computing solutions; it's also broadcast to key industry publications. For more information about the It Shipped! program visit the It Shipped Web page at http://dev.info.apple.com/itshipped.html.

To enter your Macintosh product in the database, use the form located at http://dev.info.apple.com/thirdparty/submission.html.

You must also send a copy of the product to Apple at this address:

Apple Computer, 1 Infinite Loop, M/S 301-1ES, Cupertino, CA 95014, USA.

Product	Company	Mac Only/Mac First
Aladdin Knowledge Systems	MacHASP	
Apexx Technology	PCTalk Network Adapter	
ASD Software	DiskGuard 1.5	
ASD Software	DiskGuard Remote 1.5	
ASD Software	FileGuard 3.0	
ASD Software	FileGuard Remote 3.0	
Bobbing Software	Bobbing GradeBook	Mac only
Bobbing Software	Bobbing GradeBook Pro	
Color Savvy Systems	ColorMouse spectrophotometer	
Color Savvy Systems	ColorMouseTrap color pickup and editing software	
Color Savvy Systems	SMP device profiling systems	
Comtech	256K and 512K L2 Cache	
Dayna Communications	CommuniCard Plus Ethernet/Modem	
Dayna Communications	DaynaPORT BlueStreak 10/100 NuBus™ Adapter	
Dayna Communications	DaynaPORT BlueStreak 10/100 PCI Adapter	
Dayna Communications	DaynaPORT BlueStreak 10/100 PDS Adapter	
Dayna Communications	DaynaSTAR BlueStreak 100BASE-TX Hub	
Dayna Communications	BlueStreak 10/100 Bridge	
Dayna Communications	CommuniCard Data/Fax Modem	
Dayna Communications	CommuniCard Ethernet Adapter	
DynEd International	Dinami Japanese	
DynEd International	Dynamic English	
DynEd International	Dynamic Japanese	
DynEd International	Espanol Dinamico	
DynEd International	Firsthand Access	
DynEd International	Interactive Business English	
DynEd International	Let's Go	
Eloquent Systems	Eloquent Toolkit	
Ergosoft	KIDS BANK	Mac only
Ergosoft	MOGURA-TATA-KEY	Mac only
Ergosoft	Osirase-Alarm	Mac only
Font Source	Signature and Logo Fonts	Mac first
France & Associates	HomeSchool Planner	Mac only
FWB Software	Hard Disk ToolKit	Mac first
Gefen Systems	CDJ Pro	
Gefen Systems	M&E Professional	
Gefen Systems	Touch the Music	
Gefen Systems	TSE and ADB SERIES Monitor and Keyboard Extenders	Mac only
gonet communication GmbH	golive New WYSIWYG HTML Editor for the Mac	
Gryphon Software	Gryphon Bricks	Mac only
James Renken Software	Useless	Mac first
Laminar Research	X-Plane	Mac first
Live Picture	Live Picture 2.5	
Lundstrvm Design	Touch-3D	
McQ Productions/ Software Systems	MovieTools	
Microtech International	Digital PhotoAlbum	
Native Guide Software	Spanish Native Guide	
Nisseb softwares	HTML LinkChecker	
OakTree Software Specialists	Accession	
One Click Systems	ClickMail	
Quarterdeck/StarNine	Quarterdeck Mail	Mac first
Rick Zaccone	Excalibur	
Sigma Seven Systems Ltd.	PB Serial Adapter	
SoftLinc	Trans-Linc	Mac first
Stichting Audio Ease	BarbaBatch	
Teknosys	Help!	
Unicomp s.r.l.	Web Passport	
Waterloo Maple Inc.	Maple V	♣

DEVELOPER UNIVERSITY SCHEDULE

Developer University (DU) offers a broad range of Mac OS and Newton programming instruction through hands-on classes and self-paced training products. Classes are offered in Cupertino and through selected third-party trainers.

Apple Events/AppleScript Programming
Self-paced

Creating Apple Guide Help Systems
4 days/$800
Classroom
August 19–22, Cupertino
November 11–14, Cupertino

Apple Guide Integration
Self-paced, online

Get Started Programming with Cyberdog
Online

Creating OpenDoc Parts
5 days/$1,500
Classroom
September 16–20, Cupertino
October 28 – November 1, Cupertino
December 9–13, Cupertino

System 7.5 Technologies
Self-paced, online

Macintosh Debugging
4 days/$800
Classroom
November 11–14, Cupertino

Multimedia Development With QuickTime VR
3 days/$900
Classroom
July 30–August 1, Cupertino
August 13–15, Cupertino
September 17–19, Cupertino
October 15–17, Cupertino
November 12–14, Cupertino
December 10–12, Cupertino

Newton Programming: Essentials 2.0
5 days/$995/$1500
Classroom
September 9–13, Cupertino ($995)
October 21–25, Cupertino ($1500)
December 9–13, Cupertino ($1500)

Object Oriented Fundamentals
Self-paced

Programmer's Overview of Mac OS 8
1 day/$200
Seminar
July 26, Cupertino
August 30, Cupertino
September 27, Cupertino
November 1, Cupertino
November 22, Cupertino
December 20, Cupertino

Programming With QuickDraw 3D
3 days/$600
Classroom
September 9–11, Cupertino
November 4–6, Cupertino

QuickStart Mac OS Programming
5 days/$1500
Classroom
July 8–12, Cupertino
August 19–23, Cupertino
September 30–October 4, Cupertino
November 4–8, Cupertino
December 16–20, Cupertino

Scripting With AppleScript
2 days/$400
Classroom
September 9–10, Cupertino
November 18–20, Cupertino

What Is Mac OS 8?
Online

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

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

AppleLink—Developer Support:Developer Services:Apple Developer Services:Developer Information:Developer University

Internet—http://dev.info.apple.com/du.html

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

APPLE INTERNET PAGE

This feature lists Internet resources for online information about Apple Computer; its products, technologies, and programs; Mac OS and Newton programming; and other subjects that pertain to the business of computer product development. It includes Internet resources from Apple Computer, as well as from other companies and people. Here, we list the names of the resources, and we describe new sites that we've discovered in the past month, or sites with new postings that we think you'll want to see. At the *Apple Directions* Web page **http://devworld.apple.com/mkt/adtop.shtml**, you'll find descriptions of all the sites listed, as well as links to the sites themselves. We'll update this feature every month, based both on what Apple is doing on the Internet and on your feedback.

New This Month

Apple Developer World Ⓐ
http://devworld.apple.com
Now the official Web site for developers from Apple Computer, Apple Developer World integrates technical and marketing information for developers in a single site. This is the first place you'll want to go if you're a developer looking for information from (or about) Apple, whether it's the latest news, technical documentation, software, or market studies.

Cyberdog Pound
http://www.microserve.net/~dhughes/
A non-Apple site dedicated to news and information about Cyberdog, including Cyberdog programming tips and links to sites with Cyberdog parts.

CyberTech Ⓐ
http://www.cybertech.apple.com/
Maintained by Apple's CyberTech Development group, this site includes Internet tools from Apple (including NetFinder and a guide to Web page design) as well as information about Apple's Internet development efforts.

develop Ⓐ
http://dev.info.apple.com/develop/developtoc.html
Articles from *develop,* Apple's award-winning technical journal.

EvangeList
http://www.evangelist.macaddict.com/
First there was the mailing list; now there's the Web site. Go here for archived messages from Guy Kawasaki's popular mailing list of good news about Apple platforms, as well as a variety of other useful Apple platform technical and marketing information.

Inside Macintosh Ⓐ
http://devworld.apple.com/dev/insidemac.shtml
Includes virtually the entire *Inside Macintosh* library of technical documentation online, in HTML format for easy reference.

KRT's Macintosh Business Pages
http://www.krt.com/mac/
A non-Apple page with information about client/server (and other) development tools for Mac OS systems.

MacAddict
http://www.macaddict.com/
The Web page for *MacAddict,* a new magazine for Macintosh enthsusiasts.

Power Computing
http://powercc.com
Contains information about Power Computing's line of Mac OS–compatible computers, including the 225-MHz PowerTower Pro system.

QuickTime Ⓐ
http://quicktime.apple.com
Go here to download QuickTime 2.5, the latest version of Apple's multimedia authoring architecture.

World Wide Web User Survey
http://www.cc.gatech.edu/gvu/user_surveys/
Data from the Georgia Institute of Technology's semi-annual survey of Internet users, including a great deal of demographic information about 'net surfers.

Internet Resources for Apple Platform Developers

Locations (URLs) for the following resources can be found by going to **http://devworld.apple.com/mkt/adtop.shtml**, and clicking on Apple Internet Resources.

"Ask Apple" Tech Support FAQs
Ambrosia Cafe
Apple Asia
Apple CEO Gilbert F. Amelio
Apple Competitive Information
Apple Computer
Apple Developer Catalog
Apple Developer Relations Key Contacts
Apple Developer Services and Products
Apple Developers Listing
Apple Directions Express List Server
Apple Education
Apple Europe
Apple Forever
Apple FTP Sites
Apple International Developer Services and Products
Apple Internet Servers
Apple List Servers
Apple Media Program
Apple Pacific
Apple Software Licensing
Apple Solution Professionals Network (ASPN)
Apple Strategy
AppleTalk Network System Support
Apple Tech Info Library

Brad's WebSTAR/MacHTTP
CI Labs
Complete Conflict Compendium
ComputerWare's Macintosh Links
Cult of Macintosh
Cyberdog
DayStar Digital
Developer Depot Catalog
Development Tools
Digitool (Macintosh Common Lisp)
Electronic Publishing Risks
Firewire (IEEE 1394)
Gif.glf.giF
Gradient—DCE for the Macintosh
guideWorks
Guy Kawasaki's EvangeList List Server
Guy Kawasaki's Semper.fi List Server
Happy Puppy's Macintosh Games Page
Hartsook Letter
IBM Microelectronics
International Software Development Guidelines (unofficial)
Internet Service Providers Directory
It Shipped!
Linux for Macintosh (MkLinux)
Mac*Chat Newsletter
MacHack
Macintosh Advantage
Macintosh Application Environment
Macintosh Help Wanted
Macintosh PowerBook and Mobile Computing
Macintosh Prices, France and United States
Macintosh Programming Tools
Macintosh Speech Recognition
Macintosh Vendor Directory
MacintoshOS.com
Mac OS
Mac OS 8
Mac QC Links
MacSciTech
MacTech Magazine
Macworld Expo
Mac vs. UNIX Web Server Performance
Mercury Software
Metrowerks
Motorola Computer Group
Nathan's Everything Macintosh Page
Natural Intelligence
Newton
Nisus Software
OpenDoc
OpenDoc List Server
OpenDoc Part Ideas
Open Transport
Part Merchant Home Page
PC Fairy Tales
Pictorius
Pippin
Polymorphic E-zine
Power Macintosh
QKS
QuickDraw 3D
QuickDraw GX
QuickDraw GX Fan Club
QuickTime
QuickTime Live!
QuickTime VR
Quinn's Human Interface Subtleties
Shareware.com
Shownet
Software Publishers Association
Software Unboxed
Speech Technologies
Technotes
Third-Party Products
TipWorld
Ultimate Macintosh Page
User Group Connection
Web Promotion Services
Yahoo

You can find locations (URLs) for the above resources by going to **http://devworld.apple.com/ mkt/adtop.shtml** and clicking on Apple Internet Resources. ♣

Internet Resources for This Issue

News

- Apple Developer World site—http//devworld.apple.com
- Apple Web site—http//www.apple.com
- Entry forms for Macintosh Product Registry—http://devworld.apple.com/mkt/thirdparty.html
- Apple Developer Catalog Online—http://www.devcatalog.apple.com
- CI Labs Web site—http://www.cilabs.org
- Georgia Tech's survey about Mac OS use—http://www.cc.gatech.edu/gvu/user_surveys/survey•04•1996/
- Kantara Development's component software—http://www.partmerchant.com/
- BuyDirect.com—http://www.buydirect.com
- Apple's OpenDoc site—http://www.opendoc.apple.com
- New Macintosh model press releases—http://product.info.apple.com/pr/library/1996/august.html
- QuickDraw 3D article—http://dev.info.apple.com/appledirections/oct95/quickdraw3d.html
- QuickDraw 3D home page—http://www.quickdraw3d.apple.com

Technology

- *develop* Web site—http://dev.info.apple.com/develop.html
- Year 2000 Web site—http://www.year2000.com
- International Language Engineering Corporation Web site—http://www.ile.com
- Gregorian calendar reference—http://es.rice.edu/ES/humsoc/Galileo/Things/gregorian_calendar.html
- Leap year information— http://www.ast.cam.ac.uk/RGO/leaflets/leapyear/leapyear.html

Apple Developer Catalog Ordering Information

To place an *Apple Developer Catalog* order from within the United States, contact *Apple Developer Catalog* at 800-282-2732; in Canada, call 800-637-0029. For those who need to call the U.S. office from abroad, the number is 716-871-6555. Or, send e-mail to APDA@applelink.apple.com. The *Apple Developer Catalog* is also available online on the Web (**http://www.devcatalog.apple.com/**).