elcome to the first Macintalk of the last year of the second millenium. Macintalk is my series of monthly interviews with people who make a difference in the Mac world and beyond.

This month the technologically sophisticated, the technologically interested, and those who are just curious should be interested in the following interview with Andrew Barry of REAL Software. REAL Software is the publisher of REALBasic, possibly the foremost implementation of the BASIC programming language on the Mac. Andrew speaks in BASIC (is it possible to dream in BASIC?), so for your reading pleasure I have translated the following interview into English.

Check out this month's interview, then email me your ideas, comments, and suggestions for who you would like to see interviewed in an upcoming issue.

Interview conducted 18 December 1998

ruce Klutchko: Please tell us about your background. How did you get into programming?

Andrew Barry: I'm a mainly self-taught programmer — I first started programming when I was nine on my father's [Radio Shack proprietary] TRS-80 Model I (with a whole 4 K). Of course my programming horizons expanded immensely when he upgraded it to 16 K of memory (and the substantial improvement in the BASIC implementation that came along with the upgrade).

Of course in my early days, my programming wasn't exactly rocket science

— a program that printed "Hello world" and kept it on the screen was pretty exciting.

As I grew older, my proficiency grew, as well as the power of the machines I was using: a MicroBee (a hobbyist computer in the Australian market — still based around the Z80 processor, though with 32 K of memory!), an Amstrad CPC (a hobbyist computer from Britain — I don't know if they ever made much of a presence in the US market — again, Z80 based, but with 128 K of bank–switched memory), and Amiga 1000 — my first exposure to the 68000 processor, and a whopping 2 MB of memory, once I added a RAM expansion.

At this point I'm just about to enter a university — to be specific, Australian National University — but one of my crowning works was a Modula-2 style compiler I wrote on my Amiga as my Year 12 final semester project. Actually, this was a pretty nice piece of work for an entirely self-taught compiler writer.

BK: Did you have formal training?

AB: I've got a Bachelor of Science degree which was predominantly Computer Science based.

BK: How did you become interested in the Mac?

AB: One of my close friends in university had just bought a IIci — we'd previously played around with doing games development (nothing completed, at least on the Amiga), but here was a more interesting machine with far more potential. Of course, my first programs didn't even use the Macintosh Toolbox — but slowly I managed to scrape together my first Inside Macintosh set, and slowly but surely started to tame this new platform.

Once again, my learning sped up substantially once I had managed to save up enough money to purchase a IIsi, just a day after they were released in Australia.

BK: How about REAL Software? Some of our readers have been following you since the Cross Basic days, but most of us would like to know more about who works at REAL Software.

AB: The other people at REAL software are, in alphabetical order: Janet Davis in Customer Service. Janet handles orders, customer service and shipping. Jannice Faber in Administration, handles the accounting, is our press contact, manages our trade show presence, as well as all of the other administrative tasks necessary to keep the company running.

Jason Parsley inTechnical Support is the first line of support — which takes a lot of the load off me. He also has writes and maintains the custom order system we use to automate our order processing. This system is the reason we can handle so many orders from start to finish without requiring a large staff. Geoff Perlman, President and CEO, is the main guy — he runs the company, works with me on product planning, and wrote most of the documentation.

We also now employ several people on contract to do our graphic design, documentation updates, set up our international distribution, etc.

BK: Who are the other people who are important to the success of REALbasic?

AB: The entire REALbasic user community is important to the success of REALbasic — without the support, and sometimes patience, of our users, REALbasic would never have gotten as far as it has — to say nothing of future versions.

BK: The Mac has not had the same choice of programming languages as the Wintel platform. Some time ago, we had Lightspeed C, Think C, and then MPW C. Now we have Code Warrior C. There are now several versions of BASIC available for the Mac, but until REALbasic, none have seemed to do very well. In fact, one BASIC with a visual interface, developed at about the same time as REALbasic, seemed to stop being supported just as it was released. Why do you believe this situation came about?

AB: Well, it's always been a big leap of faith to implement a programming language — and most corporations are much more willing to put their engineering time into writing new WYSIWYG HTML editors. So, a lot of the development work being done on development environments is being done by lone programmers in their spare time.

The reasons that a programmer wants to be writing a development environment in their own time tends to be one of the following reasons: for fun — so they know that they are able to do it, or for play — the programmer has some wild idea and they want to make it fly.

The problem with this is that there are no commercial considerations — indeed, usually just after they've reached the proof of concept, they drop the project and start a new one, because they've already proven they can do it, or because they've got a wild new idea that's even more interesting than the previous one.

I was, of course, such a lone programmer — but one day I decided that I was going to stop jumping from concept to concept, and instead sit down and finish programs. Luckily REALbasic was such a broad concept, that there is a lot of room to try out interesting concepts, while still progressing the product as a whole.

Turning a lone programmer project into a commercial solution requires a new or established company to take over the project and fill in the blanks. One classic example of this is how Metrowerks transformed itself from a marginal Modula-2 compiler vendor into a leading C/C++ compiler vendor by buying their C/C++ compiler from a lone programmer.

Of course, the whole process of a commercial company being interested in taking the risk with a product is related to the size of the niche and the competition in that niche.

The perceived weakness/future of the Mac platform has meant very few development environments coming out in recent years.

BK: Is the Mac a good programming environment? As someone who has had to look at the OS from the inside out, just how good is it compared to the competition?

AB: There's sort of a love-hate thing going on here. To compare the Windows API and the Mac Toolbox, the Windows API operates at a higher level, but this higher level nature means that sometimes under Windows you're forced to be competing with the OS to find some workaround to get exactly the behavior you want.

On the other hand, the Mac Toolbox works at a very low level, which means that there's a lot more work to get a simple app up and running, but you've got a much finer degree of control on how your program behaves.

BK: I'm surprised to hear that. So many of the older DOS programmers used to complain about the Mac, saying that they'd never program on a computer where the ability to control to computer was to such a large degree predetermined by the OS.

AB: This is largely an artifact of the "since I have chosen this OS/car/city/..., it must, perforce, be the best OS/car/city/...". I mean, they're hardly going to say "I feel like a complete idiot for continuing to support this outdated OS".

While the MacOS works at a somewhat lower level than Windows, it's still a far smaller conceptual gap than that between DOS and the MacOS. (You must remember that Microsoft worked closely with Apple on the Macintosh — inexplicably, chunks of the Windows API are very similar to the Mac Toolbox).

Once upon a time, I was a reasonably proficient DOS programmer — writing TSRs, using assembler, etc — and if you were going to write even a semi-professional program, your main interaction with DOS, outside of file I/O, was a chunk of code telling DOS to get out of your way (in the form of critical error handlers, etc). I mean, no-one in their right mind would use the BIOS for keyboard input, screen output, serial port control, etc. — it was such a POS that you bypassed it wherever possible.

Of course, I'd love to have memory protection — can't wait for Mac OS X...

BK: As a developer, how do you see OS X changing the user experience, both for the average Mac user and the kind of user who will purchase REALbasic?

AB: The changes will be subtle but actually profound for the user experience — I mean, putting aside any of the visible changes that MacOS X provides, the addition of preemptive multitasking between programs provides a far smoother user experience. For example, the first time I tried Windows NT after using Windows 3.11, while everything appeared identical, the environment was more productive for me — because instead of various programs locking up the computer or using too much processing time, I could just seamlessly switch to other applications and continue working on other bits.

BK: Do you see Apple continuing to be a viable platform from the perspective of the quality of the operating system and hardware? And how about from the perspective of its commercial viability?

AB: Yes — I believe Apple will continue to survive, even grow, in the future, and REALbasic is planning on supporting Apple for the foreseeable future.

The main thing that will contribute to Apple's success is its ability to maintain its newfound commercial sensibilities. From my perspective, Apple has, until recently, worked like a bunch of 'lone programmers' — each little subgroup has been off and doing their own little wild concept projects, while the platform, as a whole, became neglected because nobody was "taking out the trash."

This changed only when Apple started saying, "well, even though it would be more fun to develop our own (insert: graphics accelerator, expansion bus architecture, serial port replacement, memory architecture, network architecture, internet replacement, web browser, OS kernel, etc) from scratch, it would be cheaper, more stable, interoperable, faster time-to-market, etc. if we just licensed the technology from somebody else."

BK: Who is the typical Mac user who might use BASIC? Is this person the same as the person who would write a program in C? Someone who might use AppleScript? Or a (now orphaned) HyperCard programmer? When should the user customize a database such as FileMaker Pro, and when should he or she tackle a programming language? Which is best for which?

AB: "One ring to rule them all and in the darkness bind them."

The truth is that a person should always use the tool appropriate for the job. If all a person needs to do is some simple automation of another app, then using AppleScript alone is probably better than using REALbasic.

To make a very broad generalization, existing Macintosh development tools fall into two categories: low level tools, such as CodeWarrior, where anything is possible, but with a very steep learning curve, and high level tools, such as Director, are far more accessible, but are limited when you try to do things that the developers didn't intend.

REALbasic attempts to span the gap between the high level tools and the low level tools, in that it tries to be accessible to a wide audience, while having sufficient breadth to allow the user to accomplish a wide range of programming tasks.

Of course if the task you're trying to accomplish fits well within the bounds of an existing high level tool, such as Director, then you're probably better off using the specialized tool, rather than the Swiss army knife.

Taking your example of FileMaker Pro — FileMaker Pro is an excellent product, and if the user's problem is best solved by using FileMaker, then that's the solution that should be used — but if FM doesn't provide the degree of flexibility that they require, then they should look for another solution.

As an aside, REALbasic 2.0 is currently under development, which will provide strong database connectivity, including a royalty-free single-user database engine.

BK: It takes a beginner just minutes to write the "Hello World" program in most BASICs. How steep is the learning curve for the new REALbasic user?

AB: The learning curve is equally gentle, if not more so. When REALbasic opens, it automatically opens the default window — just drag a static text control onto the window and type Hello World. Then run it.

REALbasic contains a fairly broad range of features, but the amount of knowledge required to use any particular feature has been kept fairly shallow. While REALbasic is object-oriented, the user doesn't have to worry about learning OOP, unless the code they'll be writing is object-oriented.

BK: How steep is the learning curve for the experienced programmer?

AB: The experienced programmer should find it easy to get up to speed with REALbasic — the core language will be very familiar, and they'll get up to speed on the various features quickly. Programmers coming from Visual Basic on the Windows side will feel right at home.

BK: The term "Object Oriented Programming" seems to scare away some would-be programmers. How does OOP impact on the programming experience?

AB: Initially, the user doesn't really have to worry too much about object-oriented programming — at the simplest level everything is automatically handled for them. They will need to learn OOP if they want to fully exploit the program, but many users get along just fine using just what has been provided.

BK: Your program can run Visual Basic source code. Does this code run as well on the Mac as on a PC?

AB: Yes. In fact it should run even better — REALbasic was designed to be a compiled language, while Visual Basic is still mainly interpreted.

BK: Does this mean we might see some Windows programmers porting their applications to the Mac?

AB: We're already seeing this — we get lots of mail along the lines "we have this VB application and we're getting lots of interest from some of our Mac clients, and we heard that you were the best way of porting VB programs to the Mac".

In fact, with the RB 2.0 Pro feature of transparent cross-compilation to Windows, we've got some users who are planning to drop VB entirely and do all of their development in REALbasic — whether targeting Mac and Windows, or sometimes even if they're only going to be targeting Windows.

BK: It would be quite a pleasure to see the Mac helping out its unwanted offspring, Windows! Much has often been made of the difference between BASIC and C. Some claim that BASIC is a high level language, and therefore cannot have the degree of control of the Mac and the speed of execution of C code. They say that this means that "serious" programs can't be written in BASIC. How does REALbasic stack up in this regard?

AB: There's actually a couple of different strands to this question, so I'll try and address each of the components. Purely from the language standpoint, REALbasic offers several advantages over code written in C: true dynamic-length strings (including DBCS support), array bounds checking, garbage collection, and pre-emptive multitasking. All of these things require a certain amount of overhead, so in that respect, C code will typically execute faster than the equivalent piece of REALbasic code. Of course, a piece of C code is much more likely to crash — and if you hand-code into the C code the equivalent safety features, the speed much more closely approximates the speed of the REALbasic code. REALbasic version 2 has some options that allow the overhead to be reduced to some extent, but at least until version 3, there will be a slight speed disparity. Also, REALbasic offers a rich environment of windows, controls, and classes that encapsulate the commonly used Mac OS Toolbox services. While these offer lots of features and capabilities, it's certainly true the sometimes people need a finer degree of control than a 'one-size-fits-all' implementation can provide.

The traditional way a high level environment gets past limitations, whether in speed or in features, is through a plugin architecture, and REALbasic is no different in this regard. REALbasic supports its own plugin format, which offers the best level of integration. Plugin writers can write new methods, classes, and controls which seamlessly integrate into the product. Hypercard has a popular plugin extension using XCMDs/XFCNs, which people used to add many features to Hypercard that it didn't intrinsically support. REALbasic programs can also use XCMDs/XFCNs. Apple introduced the Code Fragment Manager for the PowerPC, which was about their fourth attempt at a shared library facility. REALbasic programs can directly call into CFM shared libraries, which also includes the entire Mac Toolbox. While REALbasic has intrinsic routines for constructing AppleEvents for inter-application communication, sometimes it's a lot easier to write an AppleScript — which can also be easily utilized by REALbasic programs.

BK: Plug-ins and XFCN's are very important to the viability of a programming language. What are some of the more useful XFCNs that you have seen for REALbasic?

AB: We support our own custom plugin format. Some of the more interesting plugins that have been written include Additional toolbox support, Quickdraw 3D support, and Java support.

While it's true that people probably wouldn't be writing Photoshop entirely in REALbasic, it's entirely conceivable that they could write all of the image processing stuff in a plugin (written in C) and do up the entire interface in REALbasic.

BK: Does the smaller user base for the Mac detract from the economic incentive to write a Mac program?

AB: Well, it depends. You should never really judge platforms by the entire size of the user base, but by how many copies you think you can sell. (Yes, I know this is cold-hearted capitalism, but if you want to be successful...)

So in my mind, the Mac market was a lot more attractive than the Windows market, because the product would have a far greater possibility of success in the under-subscribed developer tools market — while we would have been completely overshadowed by products such as Visual Basic on the Windows platform.

Generally, I don't think that writing programs in REALbasic should restrict them from deploying on other platforms — the 'pro' version of REALbasic 2 will contain a fairly transparent cross-compiler for Windows, and you should expect to see more compilation targets in future versions.

BK: I recently acquired a very useful freeware program, called "Sigerson," which is a program which helps in creating Sherlock sets under Mac OS 8.5. I was immediately impressed with how well it worked and its truly delightful interface. It wasn't until using the program for a while and checking the "About" box that I learned it was made with REALbasic.

AB: People who talk about how "serious" programs can't be written in a high level language are usually just chest-beaters who feel that the only way to write something is by doing it the hardest way possible.

If we're thinking of the same article, the columnist's opinion was something along the lines of "high level tools encourage amateur programmers to write software." He rationalized this with a set of the standard arguments, including: "To be honest, I don't want just anybody making software. If you don't know what a guard page is (...), then you shouldn't make software for anyone but yourself and a few friends." "If you want something done right, you do it yourself. You don't let some application make buttons for you, you do them yourself. You don't drag and drop a pop-up menu, you make it yourself."

Personally, I'm all for people expressing their creativity — perhaps Sigerson would never have been created if REALbasic didn't exist.

BK: REALbasic has a large and very active community of users. There seem to be very many message boards where users swap advice and code. How important are they to the success of REALbasic? Do you get involved with them at all?

AB: The active community of users is extremely important to REALbasic — as they are our evangelists, as well as providing a lot of self-help for the community. I try to maintain a presence on the main REALbasic lists — both to answer questions that nobody else can, but I also lurk a lot, seeing where people are running into limitations, and keeping a list of future enhancements to the software.

BK: Most people who have programmed in BASIC have encountered a code monster, called Garbage Collection, that could stop a program dead in its tracks for no apparent reason. It has been a problem for BASIC but not for other languages. Why does BASIC uniquely have this difficulty? How does REALbasic handle garbage collection?

AB: Well, this was more of a problem with BASIC implementations running on those old slow machines, where BASIC would probably be the only language that would have the temerity to attempt to have garbage collection — and the style of garbage collection used would require periodic scans of the entire heap to flush out all of the unused objects, which caused the hiccups that you remember.

REALbasic uses a progressive garbage collector, which causes the expense of garbage collection to be spread out evenly across the entire execution time — removing those irritating hiccups.

BK: One last question. As a developer, how do you judge Apple's attitude towards its users and developers? Are they making an effort to make your work easier? How could Apple do better to increase the vitality of their product?

AB: I think Apple is finally listening to its users and developers — you can judge this by the directions that their product lines and operating systems are moving.

As for ourselves, we have very good ties with various people within Apple. We're committed to making REALbasic work with many new technologies, since what good is technology if it's unapproachable?

I can't offer up a magic recipe for the success of Apple — I believe in their current strategy, and will be backing up that belief with solutions — something that I couldn't say for many of their previous strategies.

hanks, Andrew, for spending some of your valuable time with us. Those of you who want to check out REALbasic can visit REAL Software's website at http://www.realsoftware.com/ . You can also visit the Newbies' Unoffical REALbasic website at http://members.xoom.com/dataphile/Rb.html . Don't forget my personal favorite, the REALbasic Mailing List for real newbies at rb-pro-g@powermac.worldsite.net .

If there is a well-known person in the Mac community that you are dying to learn more about, please let me know. We would love to hear your suggestions and your comments about my column.

Bruce Klutchko
bruce@applewizards.net