Apple Manuals Apple][Pascal Macintosh SwyftCard Canon Cat

Jef Raskin Information

DOCUMENT TITLE

MACINTOSH PROJECT GENESIS AND HISTORY
FEB. 1981

SOURCE

APPLE FEB 1981

DOCUMENT NO. 17

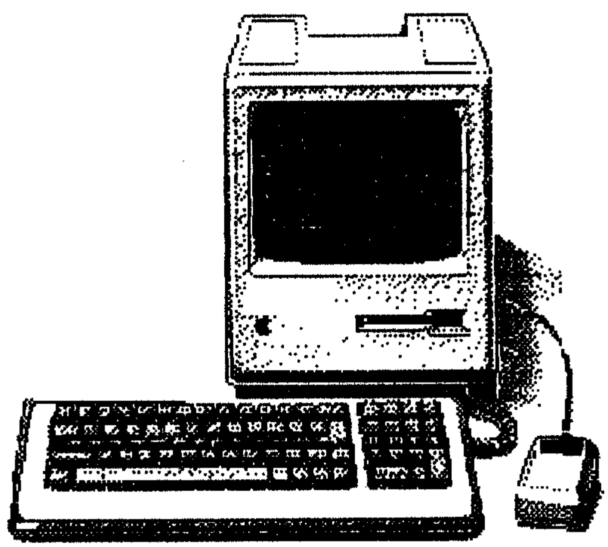
Apple Manuals Apple][Pascal Macintosh SwyftCard Canon Cat

Apple Macintosh Technical Information

Jef Raskin Macintosh Project Note

The Genesis and History of the Macintosh Project

16 February 1981



THE GENESIS AND HISTORY OF THE MACINTOSH PROJECT

Jef Raskin, 16 Feb 81

INTRODUCTION

This is a short history and background of the Macintosh research project on the eve of its becoming a product. It is offered as a contribution to our company archives, and as a demonstration of how Apple encourages innovation and supports creative thinking.

THE BACKGROUND OF THE DESIGN

I joined Apple Computer as a consultant in 1976, to write the Integer BASIC manual, and then as manager of the Publications department. Since then I started and managed the New Product Review and Applications Software departments. These departments are now scattered among our various divisions.

Writing manuals is a very special and priviledged task in a computer company, for in the process of writing them you are forced to go over every detail of the hardware and software the company sells in an attempt to make it understandable and usable to our extremely broad customer base. In the process a consciencious writer will discover nearly every good and bad feature of the system, and can provide valuable feedback to the designers and implementers.

It became very clear to me that Apple's products were, in many ways, difficult machines to explain to customers. For example to quote Brian Howard's wonderful memo "Too many variations on a theme" (30 June 1980):

> In writing [a manual] I have, once again, come across an exasperating problem: just what IS an "Apple Computer"? Apples now come in a bewildering variety of flavors.

He then points out that an Apple II behaves differently depending on what peripherals are inserted, what ROMs are on the main board, what PROMs are on the disk controller, how much RAM is installed, which keyboard is supplied, and so on. After a carefully detailed and specific list of problems, he concluded with this remark:

> The question is, for exactly what Apple, or which of the possible Apples, do we describe procedures in our manuals? Are we going to have twelve footnotes after every mention of the RESET key?... Will every writer be provided with not one, but seven or a dozen Apple systems so that descriptions of each procedure can be tested against the actual behavior every user

may experience?...

In our present context the question is, how can a naive customer deal with such a system? The answer is that he or she cannot. Even the manuals for just one sub-system, the language card, themselves require a manual to act as a roadmap to the other manuals! From this viewpoint, the Apple II/III system is already lost. We cannot go back and simplify, we can only go forward.

Considerations such as these led me to conceive the basic architecture and guiding principles of the Macintosh project. There were to be no peripheral slots so that customers never had to see the inside of the machine (although external ports would be provided); there was a fixed memory size so that all applications would run on all Macintoshes; the screen, keyboard and mass storage device (and, we hoped, a printer) were to be built in so that the customer got a truly complete system, and so that we could control the appearance of characters and graphics.

This item of controlling appearance is quite significant: for example it is impossible to write a program on the Apple II or III that will draw a high-resolution circle since the aspect ratio and linearity of the customer's TV or monitor is unknown. You can probably promise a closed curve, but not much more. You cannot promise readable characters, either. Therefore, a predictable, documentable system must be entirely under Apple's control. LISA is Apple's first system to allow us to design in context, without depending on chance for the all-important visual aspects of the computer's output.

At the time I was trying to start this project there was a growing feeling within the company that we needed a lower-cost product than the Apple II. I was originally requested to design a machine that would sell for under $500. Even hasty estimation of the cost of any practical (non-toy) system showed that goal to be unrealistic, and a new goal of $1000 was settled on. Such a goal was feasible, but the company's insistence on a number of features, such as restricting the choice of mass storage to a floppy disk rather than some sightly riskier but potentially less expensive possibilities, led to a system that will probably be offered to our customers for about $1500. Note that this leaves room for a truly low-cost system design at some future date.

In my first conversation with Steve Jobs and Steve Wozniak, back in the garage that was the original location of Apple, I argued that the Apple I (and later, the II) needed upper- and lowercase on both keyboard and screen. At the time they disagreed rather strongly--a position they now somewhat regret having taken. Similarly, when I first proposed the Macintosh architecture to the company there was strong opposition, from many levels, to the hardware and software concepts behind the machine. It is to this company's credit that they tolerated these disagreements from the beginning, and that they had enough faith in me to give me some resources with which to pursue the design to the point where it had an opportunity to prove itself.

I am honored that the company should choose to put my concepts into practice as a product.

APPLICATION AREAS

With its excellent character display (25 lines of 96 characters each) Macintosh is a natural for word processing. Given its low cost and small size it will carry word processing into areas that it was not possible to sell to with larger and more expensive equipment. If a truly portable version is produced, then an entire realm of new application areas opens up. As one small example, consider the difficulty of supplying a classroom with conventional personal computers--the problem of power cords alone is enough to be discouraging, since rooms must be specially prepared to put a computer on each desk. It does not take much imagination to see that a portable computer will open up entire new application areas, and once again give Apple access to a totally untapped, yet ripe, market.

The original concept gave the word processing program access to calculator ability without having to leave the word processing environment. Studies have shown that having a multiple level system is more confusing than a single level system. IBM's Displaywriter has a similar but more primitive facility. This opens the way to office computation applications, and a further enhancement was proposed (at the instigation of Steve Jobs) to give the editor abilities similar to Personal Software's Visicalc, except that the facilities would be embedded in the editor so that no file shifting would have to take place to use Visicalc results in a document or vice versa.

A third application area is seen in the field of distributed data bases and personal communcations. Access to a wide range of data bases could be implemented by providing a common interface and billing procedure to a number of separate commercial data services. Personal communications could be improved by a message forwarding system implemented on the user's own computer (the "completely distributed network" concept) without having to have any central computers at all.

These applications alone could result in a very high level of sales.

THE EVOLUTION OF THE HARDWARE

From the first, considerations of cost and size restricted us to a black-and-white screen. Initial mock-ups showed that a complete system in a single package could be made small enough to fit under an airline seat, and since it was self-contained could be truly portable. Power consumption estimates showed that battery power was a possiblity, although at a 2.5 kilo weight penalty. One interesting feature was the concept of a "bus diagnostic port" which would allow dealers to merely plug an Apple II (or perhaps another Macintosh) in to more thoroughly diagnose an ailing machine than would be possible with built-in diagnostics. Such a port would also allow attachment of high-speed peripherals such as a hard disk.

Another key concept is "think small". We have not begun to reach the limits of what can be done with 64K bytes of memory and a single mass storage device. It is important to hold to these limitations in order to keep the project from burgeoning into a huge, expensive and time consuming effort.

The original processor was the 6809E, but when Burrell Smith showed how a single row of RAM could be used efficiently with the 68000 processor, the machine was redesigned so that we could share the processor and much development software with the LISA project. This increased the speed of Macintosh system development, and allows us to leverage our software expertise across a range of products, much as the common 6502 basis of the Apple II/III unifies that family of machines.

The screen resolution is 384 dots across by 256 dots vertically, which is close enough to the TV standard 4:3 aspect ratio so that little screen space is wasted if the monitor is adjusted so that dots are on a square grid. Human factors considerations show that the ideal size of such a screen is very close to the viewing area of standard 9" television CRT. Bandwidth considerations limited the 6809-based system to 256 by 256 dots, which would have been optimally displayed on a 7" tube.

With a special font developed for the project, an average of 2400 easily read characters can be placed on the screen, on 25 lines of typically 96 characters each (the font, except for numbers, is proportionally spaced). This makes it very suitable for word-processing. The entire screen can be updated in about .3 second.

The keyboard's design is based on office typerwriter layout, with the backspace moved to a much more convenient location since it is so often used in word processing. The company is considering making the keyboard design standard across a number of present and future products--which would, in the long run, prove effective from many points of view such as cost, user acceptance, ease of documentation and uniformity of software design.

SOFTWARE

The design of the software, like the design of the hardware, stems from the immense frustration inherent in trying to work with existing computer systems. The best of these is probably the Alto hardware and software available at Xerox's Palo Alto Research Center (PARC). Having been associated with PARC, I repeatedcalled Apple's attention to the kind of thinking going on there, and it was gratifying that the company took note of and eventually based a lot of the LISA software on the published work done at PARC.

My concepts in designing the software were extreme ease of learning, rapid (and thus non-frustrating) response to user desires, and compact and quickly developable software. Key elements in designing such a system are freedom from modes, the elimination of "levels" (e.g. system level, editor level, programming level), and repeated use of a few consistent and easily learned concepts. Such software also leads to simple and brief manuals without having to sacrifice completeness and accuracy.

The editor is similar to the LISA editor (and thus is similar to PARC's) but does not require the expensive mouse. A careful study showed that it is probably faster to use than a mouse-driven editor--although it is probably not as flashy to see when demonstrated in a dealer's showroom. A calculator-like facility is accessible from the editor, as is the ability to link quantities within a document which gives the user an ability similar to that of Visicalc.

The calculator is modelled somewhat on programmable calculators and thus allows some programming to be done by the user, without having to learn a programming language. Its goes beyond the programmable calculator in that the user can work with a collection of numbers (such as a price list) as easily as with a single quantity. Fortunately, a long existing language, APL, has shown how this may be done so that little new work is required.

The editor can also receive and send text to external devices, including a auto-answer/auto-dial modem or a printer. In conjunction with the real-time clock it can do schedule reminding as well as automatic call-forwarding and receiving.

SUMMARY AND FUTURE DIRECTIONS

Macintosh is designed as a stable base for our low-end product line. By limiting expandability we will have a product that will be free to grow to new heights of mass marketing--for which it is essential that all Macintoshs be identical (the secret of mass marketing of software is having a very large and extremely uniform hardware/software base).

While it seems that current production plans for Macintosh may violate some of the basic principles outlined here, Mike Scott and Steve Jobs have agreed (in the Apple spirit of leading technology instead of following it) that I will continue designing and implementing the software design to the point of being able to demonstrate some of the software concepts discussed here so that the company can evaluate them.