Some Comments about Developing Applications for the Apple Macintosh 128 Computer from a 20 Year Perspective

David T Craig
941 Calle Mejia # 1006, Santa Fe, NM 87501 USA
shirlgato@cybermesa.com • (505) 820-0358

09 January 2004

o OVERVIEW

This commentary contains my recollections about developing 3rd party software for the Apple Macintosh computer during the years 1984 to 1986. This paper was written as my contribution to the Macintosh's 20th anniversary celebrations in February 2004.

During this time I worked for a small company in Wichita Kansas called PPP Inc. (PPP or P3 = Programs for Professional People) which developed a new Macintosh application for the stock market named The Investor. This application was written in Lisa Pascal and contained around 50,000 lines. I and another individual designed the program, I implemented it.

The original Macintosh (called the "Macintosh 128" since it had 128K bytes of memory) provided a fascinating development and application environment which I enjoyed immensely. The Macintosh's rich ROM-based software toolbox in a 64K byte ROM along with the machine's small footprint and superbly clear screen display made the Macintosh a wonderful application platform.

o SOFTWARE DEVELOPMENT USING THE LISA WORKSHOP ENVIRONMENT

Macintosh development in the early days (circa 1983-1985) was done using the Apple Lisa computer and its Lisa Workshop development environment. I originally used a Lisa 2/5 model which contained 1M byte of RAM, an internal 400K 3.5" Sony floppy drive, and an external 5M byte ProFile hard drive (yes, 5M as in mega bytes was considered a rather large drive in those days). I later used a Lisa 2/10 model which had an additional 10M byte internal Widget hard drive which gave me a total of 15M bytes of hard drive storage.

The Lisa Workshop hosted a command line interface which accessed a wonderful mouse based editor, a Pascal compiler, a 68000 macro assembler, an object file Linker, the RMaker resource compiler utility program, and the MacCom Lisa-to-Macintosh utility communications program.

The Lisa Pascal language was very powerful and compiled Pascal source files to Motorola 68000 object code files. I never found a need to use the Workshop's 68000 assembler since everything I needed for my application could be written in the higher level Lisa Pascal language. Macintosh application resource information was created as text files which were then compiled to a binary format using the RMaker resource compiler. Transferring a Macintosh object program from the Lisa to the

Macintosh required the Lisa utility program MacCom which copied Lisa files to a Macintosh formatted disk in the Lisa's 400K internal disk drive. MacCom combined seperate Lisa data and resource fork files which were stored on the Lisa's hard drive and stored them as single documents on the Macintosh floppy.

Macintosh programming was based on a collection of programming libraries called "units" in Pascal parlance. These resided on the Lisa and implemented the Macintosh application programming interface (API) called the Toolbox and Operating System by Apple. These libraries came on Lisa formatted disks called the Lisa Macintosh Supplement. I recall receiving around 3 or 4 supplements each with around a half dozen disks with these libraries. These disks also contained Macintosh utility and sample applications such as the Uriah Heap desk accessory by Andy Hertzfeld (called desk ornaments in the early days), the Edit text editor, and the File application by Cary Clark which showed detailed examples of Macintosh programming.

Macintosh development using a Lisa 1 model was also possible though I never worked with the Lisa 1. For this computer, which did not contain an internal Macintosh compatible 400K byte floppy drive, you transmitted you Macintosh program from the Lisa to the Macintosh via the Lisa's serial using a special Lisa utility. A special Macintosh utility received the transmitted file.

Macintosh development was also done using the Lisa Monitor development environment, but I never used this (I actually did play with the Monitor one time but thought the Workshop was a better environment). The Monitor was the Workshop's predecessor and was also command line based though its command structure was more UCSD p-system based then the later Workshop command structure. I was told that some people at Apple preferred the Monitor to the Workshop since the Monitor compilation and assembly was faster (specifically, Bill Atkinson and MacPaint and the Macintosh Finder team, Bruce Horn and Steve Capps). I've also seen Workshop references in Apple Macintosh source code to the "porkshop" but think this was somewhat unfair.

Around later 1984 or 1985 Apple provided the Macintosh Development System (MDS) which ran on the Macintosh 512K model I recall (I believe the Macintosh 128 couldn't run MDS but may be wrong). This allowed you to develop 68000 assembly language programs on the Macintosh. I never used it since I didn't write in assembly language (too tedious) and I had a Lisa with the Workshop and its wonderful Lisa Pascal language.

Concerning the Macintosh Plus computer which debuted in 1986, this computer was the last Macintosh whose system software was developed by Apple using the Lisa computer and its Workshop environment and the Lisa TLA 68000 assembler. Future Macintoshs were developed using Apple's MPW environment.

Note that the Lisa Workshop also supported a C compiler around 1985, but very little Macintosh development used Lisa C.

o SOFTWARE DEVELOPMENT USING THE MACINTOSH MPW ENVIRONMENT

I used a beta version of the MPW (Macintosh Programmer's Workshop) programming environment around late 1985 early 1986 for Macintosh development. This was Apple's successor to the Lisa Workshop which was being discontinued since the Lisa hardware had been discontinued in 1985.

MPW ran on the then new Macintosh Plus computer which contained 1M byte of RAM and an internal 800K byte 3.5" floppy drive. I recall using an external floppy drive for my MPW development of The Investor application which worked fine, but compiles were much slower than the Lisa Workshop compiles.

MPW was a very good development environment which I still use today (it now is up to version 3.4 or 3.5 I believe).

o MACINTOSH TOOLBOX AND OPERATING SYSTEM API

Macintosh programming was based on the Macintosh application programming interface (API) called at that time the Macintosh Toolbox and Operating System routines. There were around 500 of these routines in the original Macintosh. As a comparison, I just counted the number of routines in the Macintosh API MPW 3.2 Pascal interfaces from 1990 and there were around 2,300 routines (almost 5 times as many).

The Macintosh API introduced (at least to me) new programming topics such as event based programming, resources, and internationalization of text, numbers, and dates.

One idea that the Macintosh API attempted to teach developers was that the Macintosh was really a software system and not a hardware system. Prior Apple systems (the Lisa excluded) such as the Apple II and III families were more hardware oriented and minimal API information existed. Instead of writing data to a memory location for screen displaying, you instead used the QuickDraw graphics library. Apple wanted Macintosh developers to use the Macintosh API extensively since it already provided most of the core features of applications, ran fast, and was well documented. API usage also tended to promote a standardized user interface which really did not exist for Apple's earlier Apple II and III computers.

The Macintosh Print Manager was a joy to use. It provided a device independent architecture for printing really nice looking text and graphics. The old days of sending printer specific control codes to a printer and hoping for the best were at an end.

The Macintosh Memory Manager and its use of double indirect memory references called handles was an eye opener. This handle architecture provided a simple way to maximize the use of the Macintosh's limited memory size when memory blocks needed resizing (the Macintosh team has to thank Tom Malloy of the LisaWrite word processor team for this).

o INSIDE MACINTOSH

The Macintosh API was documented in a wonderful collection of notes called collectively "Inside Macintosh". Originally distributed on a chapter basis these eventually were collected in several volumes. Each chapter documented a specific Macintosh API "manager" such as the Menu Manager. Volumes 1 to 3 from 1984-1985 documented the original Macintosh API information. Volume 4 from 1986 documented the Macintosh Plus and the API changes made for this machine (such as the new SCSI disk manager). Volume 5 from 1988 documented the Macintosh II and its extensive API additions (such as Color QuickDraw).

The early Inside Macintosh chapters also contained API features which were later removed by Apple. For example, the Core Edit manager supported styled text and was a superset of the simpler TextEdit manager. Core Edit was documented in a 1982 or 1983 Inside Macintosh chapter, but was removed from the 1984 Inside Macintosh. Core Edit was used in the original MacWrite word processor.

Inside Macintosh was from my perspective very well written and provided in a very readable fashion a structure which made understanding the Macintosh API much easier. Inside Macintosh's structure was designed from the beginning and all the chapters had the same appearance and readability even though they were written by many different people. Caroline Rose was the key person behind the original Inside Macintosh chapters. She was ably assisted by around a half dozen writers.

Technical notes were also provided as part of the early Inside Macintosh releases. I recall a note from Bill Atkinson describing the internal format of MacPaint documents (he was responsible for the wonderful drawing application MacPaint, the QuickDraw graphics library, and the HyperCard user-oriented "software erector set").

Actual Macintosh system programming sources were also provided as examples. These includes all the Macintosh "definition procedures" which implemented features such as window and menu appearances (Andy Hertzfeld wrote these). The sources for the more interesting ROM managers such as the Window or Menu Managers was alas not provided (maybe today, how about it Steve Jobs?).

The User Interface Guidelines chapter was in my opinion the most innovative area in Inside Macintosh. This provided a description of the Macintosh's ideal user interface and a rationale behind the decisions.

Compared to the later book-based Inside Macintosh information that Apple produced around 1990, the original chapter-based Inside Macintosh information was for me more readable and concise. The later material tended to be wordy and overly simplistic.

o LISA MACWORKS

In 1984 and 1985 Apple supported the Macintosh operating system on the Lisa. This system was called MacWorks and allowed most Macintosh applications to run on a Lisa 2 computer. MacWorks booted the Macintosh OS from a single 400K floppy disk and even displayed the standard "happy Macintosh" boot icon. I recall MacWorks running well as long as the applications you used were well behaved (my Macintosh application The Investor was).

o EARLY MACINTOSH DEVELOPMENT DISAPPOINTMENTS

Though the original Macintosh provided a revolutionary user interface and application programming interface (API), there were some disappointments from my perspective.

Programming the Macintosh took a long time. Instead of having an application interface consisting of a simple command line interface whose output was a bunch of text lines in a fixed size font, you instead had to manage menus, multiple windows, resources, and events.

Apple could have developed higher level API routines which would have lessened some of the 3rd party development work. For example, in addition to the TextEdit manager, the Macintosh ROM would have contained a TextEdit tool which would have displayed and handled multiple text windows. Unfortunately, this would have required additional programming resources on Apple's part and possibly a larger ROM (say 128K instead of 64K bytes). This type of problem was later solved to some degree by Apple's MacApp object oriented environment but that was many years down the road from 1984.

Sophisticated Macintosh applications required more resources than the Macintosh 128 provided. The original Macintosh's 128K bytes of memory and 400K byte disk drive were on the small size when it came to sophisticated applications (I recall reading that even in Apple there was lots of discussion about this). The original Macintosh was really around a 90K byte memory machine since the screen took 22K bytes of memory and a bit of memory was devoted to system code such a ROM patches and file system buffers. I recall my Investor application was around 200K bytes in size and though it ran on the original Macintosh it was slow due to constant application code segment swapping. The Macintosh 512 ran our program well. It is a shame that the original Macintosh didn't have a bigger memory (I recall reading about a 256K Macintosh) and more disk space (the Macintosh originally used a Lisa 860K byte 5.25" Twiggy floppy drive which would have been wonderful, but in late 1983 Apple changed to the 400K byte 3.5" Sony micro-drive).

From a programming perspective, the Lisa Pascal language was good, but it could have been better. For example, routine and variable names were significant to only 8 characters. This meant that the names such as FlushBuffersNow and FlushBuffersSoon were seen as the same name, FlushBuf, by the Pascal compiler. Apple should have changed the compiler to support at least 16 character name significance, or even better 32 characters. 8 character significance was a real pain for me and reminded me of Apple's Apple II and III Pascal compilers. This naming limitation also caused the Macintosh API routine names to sometimes be very abbreviated.

Macintosh API routine names should have been named to indicate their origins. For example, I thought all Event Manager routine names should have started with EM or EM_ such as EMGetNextEvent or EM_GetNextEvent . This would have at least provided a visual clue in source listing that differentiated your application routines from Macintosh API routines. The Lisa API did this to a far better degree than the Macintosh and both used the same Lisa Pascal compiler.

Macintosh debugging using the MacsBug 68000 debugger was too low-level. I wanted a source level debugger since MacsBug was assembly language based.

Internal Macintosh API data structures should not have been published in Inside Macintosh. Apple knew these were going to change so should not have tempted developers into using this information which can cause incompatible applications when new OS versions are released. This would have been difficult to do given the Lisa Pascal compiler's scoping limitations, but Apple could have changed the compiler to support public and private information better (a Modula-2 reference mechanism could have been useful here from what I know).

The Macintosh API use of global variables was not good (these were also known as "low memory globals"). These promoted the Macintosh as a single process system which later was difficult for Apple to upgrade when it wanted to run real processes on the Macintosh. These global variables also made the Macintosh API non-reentrant which caused problems for interrupt-based tasks.

After using the Lisa and its wonderful Office System during my Macintosh development days I was disappointed that more of the Lisa's software architecture was not implemented on the Macintosh. The Macintosh was based mostly on the Lisa's visual aspects but missed other architectural elements which would have made the Macintosh a better system in my opinion. Too bad Apple could not have better leveraged off of the Lisa's best features to create a Macintosh that was really Lisa version 2 (I know the Macintosh team would cringe at this, but suspect the Lisa team would say that would have been the correct approach which was best for Apple's long term prosperity).

For example, the Macintosh should have supported virtual file names instead of file names tied directly to the file system. The Lisa finder (called the Desktop Manager) supported virtual names containing up to 63 characters even though the low-level file system supported only 31 character names. There could also be multiple Lisa documents with the same name in the same folder. The Macintosh should have also been document-centric and not application centric. Lisa users never dealt with Lisa applications directly (these were called tools in Lisa parlance) but instead always manipulated stationery pads which produced documents.

o MACCOLLEGE

Around the end of 1984 I attended a wonderful Macintosh programming seminar called MacCollege. Held at Apple's Cupertino headquarters it provided a facility with direct access to Lisa computers for development and Apple's original Macintosh technical support team.

Support people such as Scott Knaster, Cary Clark, and Russ Daniels presented Macintosh information, answered programming questions, and helped resolve bugs in your application.

I recall at the end of MacCollege signing a large piece of cardboard paper which had around a hundred names of all the MacCollege graduates (I wonder where this is today?).

o REFERENCES

Here are a list of the key materials that I used during the early Macintosh development days. I still have all these materials including The Investor source code listing and internal architecture manual.

Lisa Workshop manuals (3 volumes, dated 1983 and 1984)

Inside Macintosh manuals (3 volumes for the early days)

MacCollege class notes (around 200 pages)

BYTE magazine and its Macintosh articles (February 1984)

MacWorld magazine premier issue (February 1984)

END OF COMMENTARY