

Mac OS X and Mac Demos - The Opportunities

(Corrections 190301)

Introduction

Mac OS X looks like another milestone in the development of the Apple Macintosh computer. In this short article we look at the possibilities for Mac Demo coders to use this new technology both for the development of, and use inside Mac Demo's.

It's conception was an article on the MacDemos mailing list which discussed the advantages and disadvantages of Mac Demo's on Mac OS X. We plan to expand that discussion, although most certainly it will not become a definite list of all the possibilities. Feel free to give extra [suggestions](#) as we are willing to revise this document.

For the Developer

There are several key possible advantages to the developer, specifically the option of faster development times, and better data pre-processing facilities.

Protected memory will provide a much more stable environment. Developers probably have the most crashes of all Mac users. Microsoft (bless their cotton socks) have long recognized the advantages of providing top quality tools. Whether their advancement of their Windows 9x line's memory protection was specifically for developers is debatable, since Win 3.1 had a knack for crashing, even when you weren't developing.

Our experience is that most releases of Mac OS have been very stable (i.e. non-crashing) for non-developer use, and although it has provided a very fast and flexible platform for developing applications (compared with other operating systems), the one thing we would like to see is a protected memory environment.

A more reactive user interface theoretically means the user is less likely to have to wait for the system, and therefore more productive. Hopefully this will continue to improve as Mac OS X matures. There is a trade off of priorities for achieving this - user events go up, and everything else goes down - but as a user of an operating system it is your input that is driving it, since the computer is a tool.

Command line programs can sometimes be much quicker than graphical tools - especially when DESIGNING tools that, for example, produce data for a demo. Writing standard C i/o is fast, quick, and yields the required results. Coupled with the scripting ability of the shell (see below), you can produce data quickly. This can beat

the good but limited graphical shell frameworks. With Mac OS X, we have the best of both worlds.

Easier Porting of Unix / BSD / GNU-Linux tools. Is there a tool that you need or want to use on a Unix platform? It is easier to port these with Mac OS X. With a Linux library you can run Linux command line applications on a BSD shell, and a freeware X server for Mac OS X can't be far away.

Already ported GNU tools - for example: the Gnu C/C++ Compiler (GCC) has quite a few extra features that make it worthy of mention itself. Although C is a very flexible language, it does have various limitations in terms of implementing various techniques and optimizations, some of which have been tackled in GCC. For example, the PowerPC has a large set of registers - why can't we use a few of these for global variables? GCC has facilities in there to do it.

Another tool not to be overlooked are the BSD command lines (shells) themselves. As well as the powerful UNIX supplied commands, there is the scripting ability of these shells. Anyone who doesn't know the power of Unix scripting should take a look. The shell also has advantages over the MPW environment for building. It is also possible to call AppleScripts from the command line, and therefore setting up a whole build pattern including graphical tools. Applescript works across Aqua and Classic environments without a problem. Therefore producing a higher level of integration than previously possible 'out of the box' without the complexity of writing fragile integration tools. Along the same lines in terms of producing input text data for a demo (for example, script driven scripting of demo's), is Perl.

The Mac OS X Java implementation is also meant to be good. Whether this holds for Java Demos is yet to be seen, but it adds to the possibilities. Java is not typically seen as a "demo" language, but then again neither is BASIC, and I've seen a quite a few very good demo's developed in BASIC!

Lastly, the possibility of learning new tools and techniques has never been greater. This, for a developer, is surely an advantage?

For the Demos

Most demo coders (myself included)have felt that "Demo's (and games?) benefit themselves from OS's in only two ways: standardized hardware support and (to a lesser extent) standard library code (e.g. open gl) that is part of the standard OS. From the demo's perspective it's no good having the ability to run multiple applications, manage files, and the million other things an OS has to do."

But perhaps this needn't be the case. Features in the operating system can be used as a lever to produce a faster, better applications. This should be the case for Demo's as

well.

More APIs - the BSD heritage is available. The Cocoa interfaces are meant to be a modern object orientated system interface - and therefore should help speed development and with careful use you can ensure no "in demo" speed loss. And, of course, the carbon interfaces are there as well.

Porting Unix / BSD / GNU-Linux demo's should be a lot easier. This gives the possibility of reviewing other peoples code (always a good learning experience) whilst producing something worthwhile.

Strong OpenGL implementation. With even tighter integration with the operating system, this should continue to be a strong point in the Mac OS environment. With 3D demo's ever more popular, this is an extremely important point.

Much better multi-threading and multitasking capabilities. Even light threads have an overhead. But the trade off can be a win for the capabilities for a demo itself. Typically demo programmers who program multiple elements do so by arranging these to only run a small section of their operation. By looping around a whole bunch of "tasks", multitasking is simulated. However, some things will only break into running in small sections with a great deal of effort. This extends the time to write the demo, and makes it less attractive. Of course, there are still trade off's with threads and multitasking. Too many threads, and the demo will not run at full frame rate. (The ability to set a threads priority is also essential, obviously.). Another idea for the enhanced multitasking; running four demos (split screen) may well be easier.

Lets not also forget that a Mac OS X multi-threaded application can take advantage of the symmetric multiprocessing capabilities of Mac OS X.

Memory protection as well may offer advantage to the demo itself. For example, machine code level genetic algorithms are certainly the fastest implementation, but also the most likely to crash your machine. With memory protection and multi-threading, it raises what you can allow these GAs to do. I'm sure there are other Demo specific advantages to memory protection.

More operating system responsiveness to hardware should mean less likelihood of, for example, sound being broken up.

Another "off the wall" idea that is likely to become practical for Demos with Mac OS X is QuickTime's features. For example: The movie playback is much less likely to get disturbed as you run snazzy effects around the movie window. Perhaps couple this with iMovie, and a firewire camera for some really cool effects.

Mac OS X's better networking may also be an advantage to Demo's. I know of no

current demo that either connects to a demo server, or to other people running the same demo. Why would you want to do this? Interactive demo's may be able to provide feedback to other people running the demo. The server could supply extra data to the demo (new scripts). Community servers, interactivity, fly through 3d environments with multiple characters inside with context data from thousands of fly-throughs, distributed computing power, etc, etc.

Mac OS X is also likely to have more and more specific OS sections optimized for the altivec unit (a.k.a. the Velocity Engine). This will provide an in-built speed up to the demo, without extra work for the demo coder.

In terms of memory, Mac OS X has two positive effects for Demos. Firstly, there is the requirement that it will only run on machines with 128MB and greater. This means all Mac OS X demo's can assume more starting memory, and therefore can use more memory intensive effects. It also has a better virtual memory sub-system, meaning that Demo memory requirements probably can be relaxed in some cases.

Current Disadvantages

There are a few current areas that are not ideal.

The Debugger, GDB, is an unknown compared with Macsbug, and the PowerMac debugger, and currently it is very difficult to debug carbon applications. The first issue is really a learning issue, and the second element will be resolved with time.

There is also the issue of unknown optimization capabilities of GCC (the GNU C compiler) on the PowerPC compared with the very good code output of MrC. However, MrC was an Apple developed compiler, and so all of the techniques can theoretically be grafted onto GCC to make it a top class PowerPC compiler.

Another current disadvantage is the lack of Sprockets. Although Apple did at one point reverse their decision about not including Game Sprockets in Mac OS X, currently there is no implementation on Mac OS X. This was certainly very useful to Demo coders, but there are alternative API's and techniques. It may be sprockets, or some alternative will be available soon.

There is also concern that newer features such as modern memory management and preemptive multitasking will cause a slow down in demo's generally. There will certainly be new techniques to deal with these, such as re-prioritisation, asking the user to quit other running applications to increase performance, as well as some old techniques. In the end, time will tell. It certainly seems easier for people to write more impressive screen-savers.

Conclusion

Mac OS X provides a whole host of opportunities for Mac demo coders. It should be a very interesting scene in the coming months, as the first of the Mac OS X Mac Demos come to light.

The generated interest in Mac OS X can only be good for the Macintosh, and therefore for the Mac Demo scene.

References

MacDemos, the Mac Demo Resource page -

<http://www.multimania.com/blopblop/main.html>

Lightsoft, Mac Developers, supplier of tools, games and demos -

<http://www.lightsoft.co.uk>

Apple, THE system manufacturer - <http://www.apple.com>

This article is Copyright 2001 Lightsoft, Rob Probin and Stuart Ball.