

Apple Worldwide Developer Conference 2005 Trip Report

June 6-10, 2005

by Lawrence Peng

Directors Office

L-001

Lawrence Livermore National Lab (LLNL)

DISCLAIMER: These notes are the work of the author. They do NOT represent an official position of any LLNL organization.

Some Major Take-Home Points

Although a few bumps in the road are expected, for most general users the switch to Intel processors is likely to be a non-issue. OS X will look and feel the same regardless of which processor is in the box. I expect the majority of issues to be more application or driver specific (or plug-in specific). There are still unanswered questions regarding 64 bit on Intel Macs, especially at the high end. Although the demo and development Intel Macs are Pentium 4-based, there was never any mention of what Intel chips will actually be used. I personally expect the PowerPC will be fully supported by Apple for quite a few years since we are looking at the hardware transition to take at least 2 to 2.5 years. So if you need or want a Mac now, you should just get it now.

We should eliminate any remaining dependencies we have on Classic (Mac OS 9). While Classic works on PowerPC Macs, it does not appear that it will on Intel Macs. Even if Classic can be made to "work" on Intel Macs, we should plan on it going away.

All Java 1.3.1 dependencies should be eliminated. Java 1.3.1 is not supported on Intel Macs and will not be supported at all starting with OS X Leopard. The HotSpot C2 compiler for Intel will be coming to Intel-based Macs. Apple announced last year that Java 1.3.1 was going away in the future, and this year they specified when it happens.

The WebKit framework is now open source. This is the object oriented wrapper of the open source frameworks WebCore and JavaScriptCore (which are derived from the Konqueror KHTML open source project). Together these frameworks comprise the bulk of the web services layer of OS X (<http://webkit.opendarwin.org>).

Keynote--Steve Jobs

Steve opened by saying that "Today is an important day." There are more than 3800 attendees, which is the largest in last decade. The attendees are from 45 countries, including 38 from China and 27 from India. More than 400 entries submitted to the Apple Design Awards contest. ADC membership is 500K plus and growing

Apple retail update

Apple has 109 stores worldwide averaging over 1 million visitors per week. Over the last 12 months sold 500 plus million dollars of product

iPod update

Steve said the iPod is part of the popular culture (at least in the USA), as evidenced by being on the cover of New Yorker magazine. 16 million iPods sold as of March 2005. 76 percent market share for all types of MP3 players. The iTunes Music Store recently crossed 430 million songs sold and downloaded, and has 82 percent market share.

Steve then talked about Podcasting (iPod + broadcasting). Podcasting has been described as "TiVo for radio" or "Wayne's World for Radio". Apple sees it as hottest thing in radio, and you can download or subscribe to radio shows.

8000 podcasts and growing fast. Not just amateurs; major magazines, TV networks and other media companies doing it (e.g. ESPN, ABC, NBC, BBC, Yahoo, Newsweek, Washington Post, San Francisco Chronicle, Disney, General Motors to name a few).

The ability to access and manage Podcasts will be built into iTunes and the iPod. A PodCast directory will be built into iTunes. You can search for Podcasts via URL or through the built in PodCast directory. Steve gave a demo of iTunes and Podcasting.

Update on the Mac

Steve showed the growth rates year over year for last 5 quarters ending in March 2005. PC growth slowed from just under 20 percent to just over 10 percent Mac grew over 40 percent over the same period, with the surge starting about 9 months ago

Today Apple is releasing for download today a preview release of QuickTime 7 for Windows. Thus far a billion copies of QuickTime have been shipped over its lifetime.

This week Apple will deliver the 2 millionth copy of Tiger (via retail, Macintosh sales, and maintenance programs). Steve mentioned that to date there is over 40 Spotlight plugins, 400 Dashboard widgets, and 550 Automator actions released for Tiger.

Steve highlighted some Dashboard widgets. These include widgets to:

- search Amazon.com
- search for Business Week and CNN stories
- Countdown Calendar—Microsoft Longhorn as the joke entry
- track FedEx packages, baseball scores, and television schedules
- Rabbit Radio for listening to radio stations like National Public Radio
- Wikipedia (open source encyclopedia)
- Yahoo Traffic for updated traffic information

Apple statistics show that Tiger already comprises 16 percent of the user base. 49

percent are running Panther, 25 percent are running Jaguar, and 10 percent are still on Puma/Cheetah. Expect Tiger to be 50 percent of the user base by this time next year

Steve noted there has been 5 major release of OS X in 5 years versus 1 release of Windows. Next OS X release is named Leopard, and will ship at the end of 2006 or early 2007. This is the same as the current estimated ship time of Windows Longhorn.

Transitions

So far Macintosh in its history has had two major transitions:

The first was changing microprocessors from 68K to PowerPC in 1994. Steve described it as a good move and setup Apple up for the next decade.

The second was updating the operating system from OS 9 to OS X, which Steve described as a brain transplant which has set Apple up for the next 20 years.

Now we are starting a 3rd transition. Apple is switching microprocessor architectures from PowerPC to Intel starting in 2006-7. Steve did not specify any particular Intel chip at this time. The plan is by WWDC 2006 to have Macs with Intel processors publicly shipping. By WWDC 2007 the expectation was that the product transition would be mostly finished, and anticipate the transition complete by the end of 2007.

Apple has PowerPC products in the pipeline to be delivered. The transition will happen over the next few years, and Apple expects to support the PowerPC for quite awhile.

Why is Apple doing this?

Apple wants to make best computers for customers looking forward. Steve did mention that Apple has not been able to deliver the promised 3 GHz G5, and the G5 is not yet suitable for a PowerBook. However these issues are not the most important reasons.

Steve stated that future products they want to build cannot be built using the current PowerPC roadmap. One critical parameter for Apple cited in Intel's favor on the future roadmap was power consumption. Steve talked about a notion of "Performance unit per watt" and showed 15 units on PowerPC versus 70 units for Intel in the course of mid 2006 and beyond. There was no specifics on what comprises a performance unit.

Two major challenges to overcome:

First is to get Mac OS X running on Intel processors. Steve confirmed the long standing rumors that Mac OS X was leading a secret double life for the past 5 years for the just-in-case scenario. Every OS X release has been compiled for both PowerPC and Intel. OS X was designed to be cross-platform from the very beginning.

Steve demoed OS X on Intel, and the demo machine is based on a 3.6 GHz Pentium 4

with 2 GB DDR RAM. It is the same machine he has been using since the beginning of the keynote . He showed various apps (Preview, Dashboard, Spotlight, iCal, Mail, Safari, iPhoto, QuickTime 7 player) and all looks and feels the same as before.

The second challenge is to get 3rd party apps to run natively on Intel-based Macintoshes. Steve broke down applications into 4 major groups:

- 1) widgets, UNIX scripts, Java should just work
- 2) Cocoa using Xcode development tools should just work with some small tweaks and a recompile. Guesstimate a few days of work.
- 3) Carbon using Xcode development tools should just work with more tweaks and a recompile. Guesstimate a few weeks of work.
- 4) Carbon using Metrowerks development tools need to go to Xcode first

Xcode introduced 18 months ago. 56 percent of top 100 Macintosh developers using Xcode , 25 percent in progress of converting over, and the rest are using other tools. Xcode 2.1 for developers today after keynote.

Since the PowerPC to Intel transition will take time, Apple wants developers to create applications as Universal Binaries. A Universal Binary contains all the components an application needs to run on both architectures. When an application launches, the application loader determines the processor architecture, and loads the appropriate application components. For users, a single application binary just runs regardless of whether their Mac is based on PowerPC or Intel. Support for both processors will continue into the future while the user base migrates over to Intel-based machines.

This Universal Binary is conceptually similar to the 68K to PowerPC transition in which Fat Binaries were created which contained both 68K and PowerPC code. For a Fat Binary, the OS 9 CFM (Code Fragment Manager) and Mixed Mode Manager would decide what code was needed when an application was launched and running.

For applications which had to be carbonized during the OS 9 to OS X transition, the PowerPC to Intel transition should be much easier. This presumes developers are keeping up with the Carbon technologies (e.g. Carbon Events, HI Toolbox, etc). These technologies insulate applications from legacy OS 9 issues, endian issues, and make it easier to access other frameworks in the operating system (OS). Carbon applications reliant on legacy software shims in the OS (e.g. QuickDraw), or on specific assumptions about hardware, will require extra work in order to be made as Universal Binaries.

To show how relatively painless a Carbon port can be, Steve cited the experience of Wolfram Research porting Mathematica. The process started with a phone call the previous Wednesday night to fly out to Apple and have the port ready for demo today. Mathematica has a lot of code using C, C++, Java, etc. Theo Gray (co-founder of Wolfram Research) took the stage and said they had a working port of the upcoming

Mathematica upgrade in 2 hours using Xcode 2. Only 20 lines of source code needed to be tweaked. Gray noted this port was nothing like carbonization. It is a lot easier, and he thanked Apple engineers for giving him back his weekend.

Apple recognizes that not every application will be universal on day one. To address this, Apple will ship software for Intel processor based Macs called Rosetta. The foundation of Rosetta derives from work done by Transitive, Inc. (www.transitive.com). Rosetta translates PowerPC instructions to Intel instructions using dynamic binary translation. The result is that most PowerPC binaries can be run transparently on the new Macs. Rosetta is lightweight, and translated applications will run fast (as in fast enough). The demos given showed Word, Excel, Quicken and Photoshop CS2 (including plugins). The applications did launch and they do work.

To get developers started on making the transition, Apple has a Developer Transition Kit. It consists of a 3.6 GHz Pentium 4, OS X 10.4.1 preview for Intel, Xcode 2.1 and a Developer porting guide. These are development platforms only, and not an actual product. You can order one today and it ships in 2 weeks, but must be returned by the end of 2006. It is for Select and Premiere developer members, and costs 999 dollars.

Steve invited Microsoft (Roz Ho, Manager of the Macintosh Business Unit) and Adobe (Bruce Chizen, CEO of Adobe) to the stage. Both companies committed to creating universal binaries that can run on both PowerPC and Intel architectures.

Steve then invited Paul Otellini (Intel CEO and President) to the stage. Otellini first recounted past Apple and Intel relationships. He noted that Apple brings hardware and software engineering, design and innovation while Intel brings chip architecture, scale and Moore's law. He concluded by saying that "The world's most innovative computer company and the world's most innovative chip company finally team up".

Steve closed by saying that Apple and the Macintosh are strong now, so this is the time to build for the future and become stronger. To that end, developers should start creating Universal Binaries. Steve noted that more than hardware innovations or processor architecture, the soul of Macintosh is the OS and Apple is not standing still.

OTHER NOTES:

Below are some highlights from some other sessions I attended. As these sessions were not broadcast to the general public, I am keeping the discussion at a higher level.

Mac OS X State of the Union

The stated theme of this talk was to give the 10 best practices for building applications, with an emphasis on Tiger technologies. Items 6-10 are about user features, and 1-5 are for best implementation practices. However, with the announcement about the transition to Intel processors.....we have an extra best practice which is number 11:

Best Practice 11: be processor independent

The transition was likened to taking a car and replacing the engine and leaving all else the same. To the average driver, the car should still just work. Developers are asked to construct Universal Binaries, so that a single binary runs across both PowerPC and Intel. The Universal Binary Programming Guidelines are at developer.apple.com. How much work you need to do to create a Universal Binary depends on your application.

- Scripts and Java (shell scripts, Perl, Ruby, AppleScript, JavaScript, Automator, Dashboard widgets, Java, etc) just work as they are platform independent by design.
- Unix projects (e.g. X11, GNU, Blast, TeX) mostly cross platform so it is a recompile.
- Cocoa applications may need a few tweaks, and a recompile. Any needed tweaks are in any specific custom objects the developer has written.
- Carbon apps may vary. If no longer reliant on legacy APIs, then porting is likely mostly just tweaks and a recompile. If you have significant reliance on legacy APIs, you have work to do. Using Xcode is the easiest way to create Universal Binaries.

Best Practice 10: Create ultimate driving experience to build the best Mac application

- start with great idea and target customer
- design for Aqua and Mac OS X
- iterate on an elegant layout, but keep it simple
- adopt killer OS X technologies
- deliver high production value

Technology advancements enable experience enhancements. An example is graphics card processors (GPU). A typical GPU processed 30 million pixels/seconds in 1998. Today GPUs process up to 6500 pixels/second, which exceeds Moore's Law projection of 250 million pixels/second. The fast GPU has enabled graphics to have better fidelity, and makes it possible for graphics to be rendered in real time. Fast GPUs have enabled new applications to be constructed such as Apple's iChat and Motion.

Best Practice 9: build in navigation system which can search and find like Spotlight

- embrace and preserve metadata
- Spotlight enable your documents via Spotlight importer plugins

Best Practice 8: Pimp your ride using Dashboard widgets

- do not clutter the screen. Widgets are not meant to overlap with others.
- one widget per function and widget should communicate key information quickly
- widgets are not Aqua apps. Use Dashboard controls, not Aqua controls
- back of widget is for configuration, front of widget conveys information
- make widgets beautiful, but do not overdo the use of different colors.

Best Practice 7: Make room for everyone

- support the Accessibility API
- Accessibility API has built-in features like VoiceOver for blind and visually impaired, and ability to configure full keyboard access as an alternative to mouse operations
- making applications accessible is a legal requirement for certain markets.

Best Practice 6: Make your applications fun and compelling.

Two applications were demonstrated.

- Comic Life (<http://plasq.com/comiclife/>)
- Unity--can actually export to widgets (<http://www.otee.dk/unity.html>)

Best Practice 5: Tool it right

- need universal compiler, and typically that is GCC.
- use Mach-O binary format
 - 1) glues PowerPC and Intel
 - 2) multi-architecture by design (set GCC flags: -arch PowerPC -arch i386)
- Target architectures can be set with Xcode via checkbox. Provides Universal Binary and universal application. (includes entire bundle package)

Best Practice 4: Open it up

Fundamental shift in industry to open systems built on open standards and open source.

- avoid reinventing the wheel
- interoperability, security and reliability
- focus on where you add value

Examples in Tiger:

- SQLite--takes care of byte order
- WebCore—live repository which is part of the Web Services layer of OS X.
- WebKit--object-oriented wrapper for WebCore (now open source).
- Panthers renders HTML, Tiger edits it too
- JavaScript (scripted languages fastest growing set of languages)

Use Unix open source projects if possible

- choose from right collection of commands and libraries
- download, recompile, and share you improvements

Best Practice 3: Build from parts (component reuse)

All about object orientation

Java (good for servers, IS custom apps, and cross OS portability)

- Mac OS is the platform of choice
- impedance match with Unix
- full Aqua citizen
- Java 1.4.2 and Java 5

Cocoa

- fully leverages OS X
- minimize application code by providing many automatic behaviors for free
- highly extensible/customizable and subclassable
- exo class additions (e.g. categories) and delegations
- Endian issues handled for you
- Unicode and serialization
- native language is Objective-C

Objective-C add classes and methods support to C. Power is in the runtime which provide introspection and rapid application development tools like Interface Builder (IB).

Cocoa has consistent orthogonal API. Many basic subsystems like views, controls, panels, document architectures; and scripting architecture.

New in Tiger is Core Data (object oriented API for persistent stores which is on top of SQLite or XML files). Core Data can be thought of as the Enterprise Objects Framework (EOF) for the desktop, and it provides:

- editing, validation, undo
- provides MVC (model view controller) model
- modeling tools so that you write less code

New in Tiger is Sync Services which is object oriented API for Data synchronization.

- fine control
- conflict handling
- system data types
- highly customizable

New in Tiger is Core Image to make accelerated graphics easy.

- Many image units-extensible
- compose effects
- realtime, dynamic compilation
- GPU and CPU independent

New in Tiger is QTKit which is an object oriented API for QuickTime.

- playback, editing , export
- powerful, yet simple
- effectively allows you to have a QuickTime player in your application
- used by Apple when the QuickTme player was rewritten for Tiger

Apple secret for new applications is that most of the new apps written from scratch written in Cocoa. Some demos were shown porting FreeCell and GNU mail from PowerPC to Intel with zero changes in source code using Xcode 2.1.

Best Practice 2: Supe it Up (pertains largely to Carbon applications)

Apple has added over the years Carbon Events, HIView, HIObject, HIControls, HIArchive, etc. Moving forward will see continued improvements and additions in Carbon objects, Interface Builder (IB) integration. Developers are asked to continue to perfect their Carbon applications to help ensure they can easily be made universal.

Two possible issues

- Resources

If using Standard UI resources, then it is automatic. In the best case, the developer used IB to construct them. Custom resources need to be tweaked by hand.

- Document formats

If already cross platform, then you are done. If not, you may need to tweak by hand (especially if you need to resolve endian issues). HIArchive may be of help.

Best Practice 1: Pedal to the Metal--speed is key

Retune your applications with each release

- start with overall system
- leverage the frameworks
- leverage the developer tools for measuring and optimizing
- best to write algorithms, while last resort is to write assembly
- for vector math it is best to use the Accelerate Framework. The framework has over 4500 routines for vector math, large integers, linear algebra, signal and image processing. Optimized for PowerPC (AltiVec) now, and in progress for Intel (SSE).

Finally, an overview was given about issues Apple ran into when porting various OS X components to the Intel architecture. Most things came over fine; but there were some crashes, weird results, strange colors, strange text, bad read and writes, and strange network behavior. Most architectural differences are covered by the compiler.

One note regarding endian issues is that PowerPC returns a 0 result for a divide-by-zero operation. The divide-by-zero operation on Intel results in a total system crash.

For moving from AltiVec to SSE/MMX, Apple had some suggestions. Many program structures, operations, and capabilities are equivalent. But there are some that have not equivalent, have different semantics, and have no equivalent on the other architecture. If a missing capability is required, you need to come up with alternative. The best way is to use the Accelerate Framework when possible. One could also leverage code for other OS (e.g. windows), but recommend to start with C-based code.

Development State of the Union

New stuff for Tiger (Tiger is universal)

Automator

- natural evolution of AppleScript and script editor
- bring your application's object model to the power user
- automation is easy for users which should help to increase adoption
- easy for developers to create actions
- 500 3rd party actions as of today

Xcode 2.0/2.1

- fastest way to create app for OS X
- codesense predictive compilation, distributed builds, zerolink, fix and continue
- GCC 3.3 to 4.0
- 20-30 percent faster compiles for C++ and Objective-C++.
- tree-ssa (single static assignment)
- autovectorization 10-20 percent improvement
- Xcode project file format is readable, differentiable and can be merged.
- New format xcode to xcodeproj
- Unit testing built into Xcode 2.1. It is why Apple shipped Tiger on time.
- debugging enhancements inspired by Automator UI

Other updates include:

- Java 1.4 to 5
- Shark 3.8 to 4.1
- WO 5.2.3 to 5.2.4
- Quartz composer
- ADC Reference library

Time to broaden

- improve compile chain, debugging, profiling, other tools
- Host to multiple targets
- Universal binary one executable multiple segments
- GCC4--latest from FSF, many tune options for Intel, etc.
- shared precompiled headers

SDK (Software Developer Kit) are key to universal binaries

- compile with headers and link with frameworks for Jaguar and above
- take advantage of the newest Xcode features
- live on the newest OS, always with the latest tools
- provides release to release binary compatibility to previous OS X releases

CHUD 4.2 developer preview download

- supports Intel based Macs

- new system trace configuration
- better support for grid and cluster

Java 5 preview for Intel based Macs available for download soon The preview release will have both C1 and C2 Hotspot based compilers

WebObjects 5.3 tools now in Xcode 2.1

- Development license with every copy
- new emphasis on OS X
- EOF modeling tools integrated

What about Fortran?

Kevin Smith (Director, Intel Compiler Lab) took the stage to announce that ICF Fortran from Intel is coming to OS X later this year. Intel will support both C and Fortran on OS X, and will have more details at the Intel Developer Conference in the August.

Graphics and Media State of the Union

OS X goals were to provide pro grade audio technology, ultimate 2D graphics, hardware accelerated 3d graphics, and state of the art image processing in order to be the leader in digital media. The architecture needs to be super fast, integrated by design, simple to use and future proof. Industry moves quickly, so one must evolve or die. If developers write to the updated API's, then the effort to transition to Intel Macs should be minimal.

Typically 4000 titles every year for QuickTime licensing. It is estimated that 1 billion copies of QuickTime have been issued since QuickTime's launch. QuickTime 7 supports for Tiger, Panther and MS Windows.

Among the new stuff for Tiger is:

- H264 codec
The codec has a more complex picture and bitstream structure, and edits require much more sophisticated logic. However, your editing API remains simple and intuitive, and Apple had the headaches so you do not have to.
- High definition audio
QuickTime directly layered on Core Audio, supports many surround audio formats, sample rates up to 192 kHz, sample resolutions to 24 bit and beyond, and sample accurate multi-channel sync
- Video on a texture (common request which is hard to pull off)
The solution is Core Video, and is underneath QuickTime. CoreVideo gives ultra high performance display blitting, hardware accelerated compositing, and is the integration point for Core Image and other OpenGL processing.

- QTKit (QuickTime Kit) framework.
QTKit is a Cocoa framework to access and manipulate the QuickTime API. Apple uses it to build the Tiger QuickTime player. Another application which was built on it is Pitch Doctor 2 which was demoed by author Michael Johnson from Pixar.
- Core Image
 - Quartz 2D and PDF integration with core image
 - utilizes OpenGL GLSL (shading language)
 - Core image demo app Fun House part of developer's tools distribution
 - Core Image utilized by the Quartz composer developer tool
 - Image units in analogy to Audio Units of the Core Audio framework
- Core Video
 - optimized for video on a texture
 - forms basis of QuickTime 7 new video pipeline

Quartz 2D Extreme is not turned on in Tiger (as of version 10.4.1) due to inconsistent usage of Quartz by applications. Applications can get performance boost or deficit. Apple is asking developers to drop any dependencies on QuickDraw. Leaving QuickDraw behind is required for constructing Universal Binaries.

Java Overview

Compatibility

The Java team has fixed 2000 bugs/issues since Panther was released. The emphasis was on compatibility:

- beyond the Java compatibility kit
- unwritten specs (things not documented but needed, been there a long time)
- enhanced regression testing
- JUnit (ask developers to make JUnit test results part of Java bug report)

Beyond the desktop

- J2EE development and deploy (e.g. WebObjects)
- J2ME develop and deploy (Cell phones)

Bundled extensions in J2SE

- Java advanced imaging and Java 3d
- Webstart
- Applet support
- LiveConnect

Performance

- Fastest Java Apple has ever shipped
- Using SwingMarks, for Tiger 1.4.2 is 13 percent better than Panther, and Java 5 is 25

percent than Panther.

For Tiger

- Java 1.3.1 ships with the system.
- Java 1.4.2 is the default
- J2SE 5 for OS X Release 1 (shipped on 4/29/2005) as a software download
- at this time, use Java Preferences in Applications/utilities/Java/J2SE 5.0 if you want to set J2SE 5.0 as the default JVM.

Future

- 1.4.2 Tiger update to keep up with updates and security bulletins
- Java 5 to be available as a software update soon.
- Apple would like to make it the default, but wants feedback before default change
- keep up to date with 1.5.0_0x releases

Apple is currently in talks with Sun regarding J2SE 6 (codenamed Mustang)

Java 1.3.1 is going away in the next major release of OS X (Leopard). Move to 1.4.2 and above now. Apple needs feedback on why you cannot move.

Advantages of Java on OS X

- bundled with OS X
- up to date and consistently updated Java versions
- SDK tools integrated into OS X
- excellent documentation
- Java optimized for G5 automatically

SciMark composite index

- G4 dual 1.25 is 98
- G5 dual 2.5 = 289 (better than theoretical simulation of doubled G4)
better than twice because using fast math with 64 bit registers, instructions

Java API is built on top of OS X frameworks

- AWT built on top of Cocoa
- Java 2D built on top of Quartz
- Java accessibility on top of OS X Accessibility
- Java printing (javax.print, java.awt) on top of Print Services
- Java utility framework on top of Core Frameworks
- 3D extensions (JOGL, Java3d) on top of OpenGL

Security features

- Keychain integration--security framework now used for trust management
- one place to store certificates in Java
- works with both applets and Java Webstart

JGSS Improvements

- built on OS X Kerberos implementation
- can now find Kerberos config on OS X
- access to in memory credentials
- first J2SE to fully support MIT Kerberos

Aqua everywhere

- AWT widgets
- Swing—when routed to Aqua, Java applications get the standard Mac menu bar, and anti aliasing text by default
- extended AWT (not an OS X lock-in) which provides some missing OS X specifics.
- Recommend bundling applications as an application bundle so you can have a double-clickable application versus a messy folder.

Java and Native Integration (JNI)

- learn how to mix Java and native code
- understand how to use the Cocoa component to embed Cocoa views in your Java apps (OS X specific)

Bonjour for Java

- bundled with Tiger
- available for windows
- easy to use API
- preferences file is called com.apple.dnssd
- many uses like games, chat, collaborative computing

OS X tools

- Xcode 2 which allows Java and C/Obj-C/C++ side by side
- JAR bundler
- Ant builds
- Shark and all the CHUD tools
- Terminal applications

Enterprise tools

- Tiger Server
- Apache
- WebObjects
- use 3rd party emulator for MIDP 2.0
- Apple tools for build debug, profile, deploy
- Eclipse, JDeveloper, NetBeans, IntelliJ IDEA

Transitions--What do Intel based Macs mean for Java?

What works

- all 1.4.2 and 5 apps work on both PowerPC and Intel
- Java is running processor specific
- bundled apps, applets in WebKit, webstart apps, jar files, command line tools just work

What does not work

- 1.3.1 not supported
- Your JNI libraries need to be recompiled as Universal libraries
- Rosetta applications cannot have embedded JVM's

Hotspot compiler on Tiger

PowerPC

- J2SE 5 C1
- Java 1.4.2 C1
- Java 1.3.1 C1

Intel

- J2SE 5 C1
- Java 1.4.2 C1
- J2SE5 C2 compiler. C2 is a more intensive compiler
- C2 is about 2.3x faster than C1 on Intel

WebObjects Overview

WebObjects is Apple's rapid application development environment to develop and deploy enterprise web and Java server services and applications.

Advantages

- 3 tier architecture (data access, business logic, presentation)
- visual tools, frameworks, scalability
- "wrapper" for J2EE apps or equivalent functionality
- Automated data access which means:
 - no need to write SQL
 - database independence
 - transparent persistence

Where used

- Digital and print asset management
- higher education
- scientific technical community
- Apple (Apple store, .Mac in iCards and Webmail, iTunes Music Store)

Developing with WebObjects can be loosely broken down into the following steps:

First, JDBC is modeled with EOModeler

- EOModeler reverse engineers the database finding all tables, column and relationship
- use EOModeler for schema design, and EOModeler generates Java class files for you.

Second, use Xcode to build customer Java application logic

Third, use WebObjects builder or Webservices assistant or Interface Builder to build the presentation layer for the webservice, web app or Java app.

WebObjects builder

- graphical html editor
- visual development to build data driven web pages
- reusable components
- generate dynamic pages
- generates html or xml and isolates html and logic from the page
- manages sessions, users security
- interoperable with JSP
- localization support

Webservices

- standards based web services drive app integration
- interoperable with many languages including AppleScript, C++, java and .net

Building web services on server side.

- fire up webservices assistant
- build or consume web services
- tools for configuration and testing
- code free generation of web services from existing data assets

Building desktop apps

- via IB (Interface Builder) and direct to Java
- assistant for customizing layout
- tool for creating custom swing widgets
- Java webstart integration

Deployment options are via J2EE container or WebObjects Java application server.

The WebObjects native deployment has:

- built in clustering support and load balancing
- session management
- auto restarting
- remote config and monitoring tools

Recent Events

- release 5.2.4 with Tiger supported with 1.4.2 JVM
- WebObjects server admin plugin (integrated with rest of services in Tiger server and

server administration application)

- WebObjects 5.3 on Xcode 2.1 developer tools. WebObjects 5.3 available for Tiger

Some of the new things for WebObjects 5.3

- Xcode EOModeler plugin
 - tighter integration with IDE
 - based on CoreData modeling
- updated WebObjects builder
 - user interface gets a facelift, and new preview mode based off of WebKit
 - xhtml and html 4.0.1 support
- html/xhtml compatibility
 - tool and runtime support
 - html 4.0.1 improved
 - css support
 - xhtml compatibility
- Java collection classes support
- Updated webservices
 - Axis 1.1
 - better interoperability
- Oracle 10g support formal qualification

WebObjects futures

- developer tools updates
- continued improvements in EOModeler plugin and WebObjects builder
- better deploy and monitor
- support for J2SE 5 when it becomes the default

Two final points to note:

WebObjects 5.3 drops Solaris as a deployment platform. I am not sure if this means that Solaris is dropped entirely, or whether J2EE containers work fine and that the WebObjects Java application server support of Solaris is dropped.

There is continued interest in whether EOF (Enterprise Object Framework) will ever be revived for desktop applications. EOF is used by EOModeler to abstract databases into objects and to provide persistence. The answer is that EOF on the desktop is not under consideration. For equivalent function on the desktop, use the CoreData framework.

Cocoa Today

The goals of Cocoa are:

- simple things simple, complex things possible
- we write the common code for you, so that you can do much more

Cocoa is not an isolated box. It can interact with other subsystems. It's native language is Objective-C, which is a very thin superset of the C language. Cocoa does not provide

an API for every technology on the system. The guiding principle is that Cocoa has an API for those technologies where the Cocoa object-oriented approach provides additional benefits (abstraction, power, rapid development, performance).

In regard to the term "Performance", there were a few statements made. Framework or object-oriented does not automatically imply inefficient. Cocoa was developed on machines with 4 MB and 25 MHz processors. Objective-C is object-oriented and dynamic but has low memory and CPU overhead. Object-oriented techniques and abstractions often help boost performance where needed.

Cocoa has consistent name and calling conventions. A small number of concepts which are widely reused. Cocoa is "universal". It has been ported to 68030, 68040, RISC, and X86. It is architecture independent (opaque objects).

Cocoa and Objective-C++

- a peaceful coexistence.
- runtime systems of both languages unchanged (native abi, semantics)
- subclassing across object models not supported

Cocoa and Java (bridging to make Cocoa API's available to Java)

- for the most part transparent
- bridging of new classes is not automatic and not toll free
- apps have slower startup times and larger footprints than Objective-C counterparts
- not a lot of developers are actually using Java to program Cocoa
- moving forward, Java bridging to new Cocoa API's is no longer supported

Cocoa and Carbon interaction

- supported
 - non UI code
 - model panels in the other environment
- supported with caveats
 - document windows in the other environment
- Not supported
 - hosting views, sheets, or drawer in windows of the other environment

Tools (tightly integrated, but independent of each other)

- Xcode
 - support Objective-C and frameworks
 - class and data modeling
- IB (Interface Builder)
 - fundamental tools for Cocoa UI
 - layout UI elements and specify connections between UI objects
- Performance tools

Main Cocoa frameworks (link against Cocoa.framework)

- Foundation
- Core Data (new to Tiger)
- Application (App) Kit
- there are other frameworks in the system with Objective-C hooks:

- Cocoa Bindings and Core Data
 - speed up development process
 - more functionality for less code, so you focus on the core of your app

Bindings

- based on MVC (model-view-controller) and key-value pairs
- automate synchronization between view and data model objects
- powerful controller classes to manage model objects
- bindings often configured code free in IB
- bindings easily coexist with traditional glue code, so you can mix and match

Core Data

- object-oriented API to manage data models
- makes heavy use of concepts from the enterprise database world
- fine grained object management and persistence
- lots of function for free (undo, state validation)
- 3 types of object stores: sqlite, xml, binary (key archiving)
- Object store schemas owned by Core Data

Stuff you should not be doing

- assume sizeof int == sizeof void*
- assume 1 pixel == 1 point
- assume API's return things other than what they say
- drawing faster than 60 fps
- using private APIs

HPC Technology Update

Clusters the rage since about 1998. This session focused on dedicated HPC or research clusters. Such clusters have the cluster compute nodes on a private service network with a dedicated communications network, and a number of service nodes with access to the LAN. Users login into the cluster via a service node. The compute nodes run the minimal number of running services.

Some examples of clusters using Macintosh technologies are:

Virginia Tech

425 kW load to 310kW load by going to 2.3 GHz G5 Xserve from 2 GHz G5 desktops
12.25 teraflop to be number 7 on top 500, and top academic cluster

Entered production this year, and are a technology testbed to OS X

UCLA Dawson Cluster

256 Xserve G5 nodes.

0.948 teraflop as 444 on top 500.

using 1.21 TF using Dauger Research PoochPro and MacMPI

Colsa Mach5

1562 dual 2.3 GHz Xserve G5.

University of Illinois at Urbana-Champaign Turing Cluster

10 teraflop peak

640 nodes with Myrinet, for 3 million dollars

Bowie State (Bowie, Maryland)—XSEED Cluster

224 dual 2 Xserve G5 nodes

2.1 teraflop benchmark run

Technology survey

Hardware platform

- Xserve G5 and cluster node (now supports 2 GB DIMMS)
- Apple Workgroup cluster for Bioinformatics (<http://www.apple.com/xserve/cluster/workgroupcluster/>)

Tiger Server

- 64 bit memory, Xcode 2 and GCC4
- kernel tuning via launchd
- memory manager code

Hardware interconnects

- gigabit ethernet technologies
- Myrinet (2 new Xserve cluster with Myrinet in the June 2005 Top 500 list)
- Myrinet express (MX) is a new message passing system for Myrinet clusters
- SilverStorm Technologies (formerly InfiniCon Systems) and Small Tree Communications for Infiniband
- Small Tree also doing gigabit ethernet solutions/drivers (focused exclusively on OS X)

IPC middleware list continues to expand

Tools and Code libraries

- Xcode
- Accelerate framework from Apple
- Absoft IMSL math libraries, NAG libraries, IBM xlc/xlf, etc

File systems supported include NFS, AFP, Xsan, Lustre, and Blaze

Schedulers (e.g. Xgrid, Maui/MOAB, Pooch)
N1 Grid Engine from Sun

Creating Dashboard Widgets

Dashboard is a lightweight layer in the system for miniweb applications. The key word is applications, not web pages. Widgets present developers with new opportunities, and over 400 widgets are available as of WWDC 2005.

The details of the Dashboard API will not be covered here, but can be found at :
<http://developer.apple.com/macosx/dashboard.html>

Apple does have a website for hosting widgets here:
<http://www.apple.com/downloads/dashboard/>

Apple envisions three classes of Dashboard applications.

- "Lightweight utility" (e.g. Calculator)
- Application that compliments an existing application (e.g. the iTunes controller)
- Application to summarize existing data. (e.g. widget to track the weather, stocks, etc)

Dashboard consists of a server, client, and widgets. The server's responsibility is the user interface, events, and launching clients. The client bridges the server and widget via a web view. The widget application is packaged as an OS X bundle.

Widgets (except for the base set Apple provides) are downloaded to the user's home directory and run at the user's level of privilege. At of OS X v10.4.1, widgets are all or none; meaning you can restrict dashboard usage, but not individual widgets right now.

The Dashboard "security issue" was addressed in 10.4.1. The issue was that widgets could be installed into a user's home directory without user knowledge by visiting a demo website setup to do this. However, the installed widget would NOT automatically run. Users still had to manually activate it. If a widget is malformed or obviously busted, then it will not run. The temporary fix was to turn off a preference in Safari which said "open safe files after downloading".

Starting with 10.4.2, Apple will provide a basic widget management tool (for downloaded 3rd party widgets). While not meant as a hardcore security feature, it will provide warning if an attempt is being made to install a widget covertly. Facilities for deactivating and uninstalling widgets should be available.

Because widgets are web applications, they should largely just work on the upcoming Intel-based Macs. However, developers who have utilized Widget plugins to write

custom functionality will need to check their plugins for universality.

The WebKit Open Source Project

The project went online as of June 6, 2005. The website with all the gory details of how to get the code, build, test, submit fixes, etc, are at:

<http://webkit.opendarwin.org>

History of WebKit

WebKit is an object wrapper encapsulating Webcore KHTML, KWQ JavaScript core, KJS (K JavaScript), and other stuff.

2001 Safari project begins (based on kHTML from the KDE project)
January 2003 Safari 1 beta, Webcore JavaScript core
June 2003 First release of WebKit framework
October 2003 Safari 1.1 and WebKit in OS X

February Safari 1.2 2004
June 2004 preview of Safari 1.3 and 2.0 at WWDC
April 2005 Safari 2 for tiger
June 2005 WebKit open source project

Previously, Webcore and JavaScript core sources were published, but it was not easy to build or test these frameworks yourself. Some aspects of the source code assumed you knew how other closed source areas of OS X operated. WebKit was closed source.

As of WWDC the entire WebKit is open. The WebKit system interface is the remaining closed stuff, but the binary is free. And there are directions on how to build from source.

- better testing by more people, and catching regressions earlier
- more eyes on the code so anyone can study WebKit for correctness and security
- richer participation by individuals and groups
- be a part of the community that determines WebKit's future

Beyond OS X

- Convergence with KDE
- Opportunity to integrate existing projects
 - GTK+ port
 - Omnigroup improvements (e.g. work done with OmniWeb)
- potential for new projects

Introduction to Core Data

<http://developer.apple.com/macosx/coredata.html>

Core Data is a new Cocoa framework introduced in Tiger. It is a data model framework dealing with object-graph management and data persistence, which is designed for the Cocoa application model of MVC (model-view-controller). I very loosely refer to Core Data as EOF (Enterprise Object Framework) for Clients, as many ideas are similar to EOF (many of the Apple Engineers building Core Data have also contributed to EOF).

As an aside, EOF is part of Apple's WebObjects product, and is an enterprise-class object-relational database application framework. Although Core Data is not EOF, it makes heavy use of concepts from the database world to take on application data management. It is a general-purpose data model management solution. Many of the Apple engineers working on Core Data, have worked on EOF as well.

The correlation of MVC with the tools/frameworks are as follows:

View is user interface issues, which is typically addressed by Interface Builder (IB)

Controller refers to Cocoa Bindings (in the form of pre-built controller objects) that was introduced with Panther. These objects form the glue between the data model and view. Prior to Cocoa Bindings, controller code was the developer's responsibility.

Model is the application data model, and is where Core Data fits in. Prior to Core Data, the job of creating and managing an application's data model fell to the developer.

What Interface Builder does for constructing user interfaces, Core Data does for your data model. Using Core Data tools, you graphically define the application's data model. Core Data structures are accessible by your code, and like other Cocoa frameworks, you get a lot of common functions for free (e.g. undo, redo).

Core Data arranges the application model layer into a defined set of data objects in memory, and tracks and manipulates these objects in "real-time". When you want to save changes to application data, Core Data archives the objects to a persistent store for you. Core Data can save application data as regular files that users can manipulate.

Harnessing the Power of PDF

Apple likes PDF for a number of reasons:

- proven technology which meet requirements
- public specification
- platform independent
- compact and secure file format

PDF is a good match for digital paper:

- resolution and device independent
- page oriented

- searchable, can be encrypted and annotated
- ideal for printing, sharing, archiving

Since PDF is a core technology for OS X, with little effort, you can add capabilities like:

- open, display, generate, modify, and annotate PDF files
- use PDF to handle legacy graphics formats like EPS, PostScript, and PICT
- participate in PDF workflows
- utilize PDF to and from the pasteboard

Quartz 2D is the lowest level API to read, display and generate PDF. Tiger API's in Quartz even more complete (e.g. add links to PDF and access the PDF content stream). In Tiger we can read and draw PDF 1.6. PDF generated via Quartz starts at PDF 1.3

PDF is leveraged via Quartz in a number of ways:

- PDF Kit framework (new for Tiger) provides easy PDF support for Cocoa apps. PDF Kit is layered on top of Quartz, and is a suite of Cocoa classes which call through to Quartz 2D. Enhancements to Preview and Safari is a result of the PDF Kit.
- Quartz filters for post processing of PDF files
- PDF creation for all applications via the Printing mechanism
- PDF Actions (new for Tiger) providing PDF support for Automator workflows
- PDF processing with Python scripts since Panther

PDF Workflows

- useful for high level PDF work when you do not need a full blown application
- Python scripts to create PDF (since Panther)
- Automator actions to analyze PDF (new to tiger)
- PDF workflow from the the print dialog
- PDF workflow API and PDF Scanning API
- applications can execute the PDF workflow items directly

Core OS Enhancements for BSD Developers

Apple System Logger (ASL)

ASL is a new approach to creating and managing system log messages in OS X

- replacement and extension of BSD syslogd mechanism
- want to provide structured and flexible message format
- reduce log file clutter and make files easier to read

BSD syslog has priority levels 0 to 7, representing emergency to debugging situations. syslogd is the ASL server (supports both old and new ways), and has API's for sending and searching messages. There is a syslog cli tool for searching and monitoring.

ASL will receive a message which goes to syslogd server. The server interacts with a unified log message data store, but also has legacy support for traditional configuration

and output. There is a new syslog command line tool for searching and monitoring messages being written to the data store

Filter controls (Local, master, and application) to determine which priority levels are sent to syslogd, and which priority levels are saved in the data store.

- master filter is normally off and has no effect
- if master set, it overrides local filter
- applications specific filter overrides master and local filters

Default data store filter saves emergency through notice messages, and keeps messages for a week by default. You can setup cron scripts to automatically prune the data store, or use syslog -p to prune the data store manually. There will be further improvements to manage the data store in the future.

Service Management

Introducing launchd (new with Tiger)

- unification of background process management
- launch on demand to help save system resources
- going native with launchd—a simple high level IPC API.
- new kinds of launch on demand criteria
- simplified notion of dependencies
- support for user supplied jobs

Launchd uses xml instead of shell scripts

- structured data is a good thing
- easy to introspect attributes of all background jobs
- easy to change attributes of any background job
- standardized schema for control over how daemon is started

Dependencies--the old way

SystemStarter used explicit out of band dependencies. In reality this did not work.

Often one of following happen:

- dependencies overstated, understated, meaning was vague.
- dependencies assumed by virtue of other dependencies making refactoring code hard

Dependencies--the new way

- daemon writers must define their sockets in their xml plist
- launchd uses implicit inline dependencies
- we register all sockets of all daemons first
- then start all non demand based daemons
- as daemons startup, they may talk to other daemons freely
- as daemons startup, they dequeue messages when they become ready

Java Virtual Machine Exposed

Simultaneous with Tiger's release, the J2SE 5 with Hotspot C1 compiler was available as a software download (and will be a software update soon). Java 1.4.2 applications should just run in J2SE 5, but Apple is asking for reports of any back compatibility bugs.

Tiger on PowerPC ships with 1.4.2 and 1.3.1. Intel Macs will ship with the J2SE 1.4.2 C1 compiler, and introduce the C2 compiler. Java 1.3.1 will not be shipping on Intel Macs, and official support for Java 1.3.1 in OS X is gone starting with OS X Leopard.

Today, J2SE 5 is the focus of Java efforts at Apple. J2SE 5 introduces some of the biggest changes in Java history (in the language, syntactics, memory model, etc.).

The HotSpot JVM provides

- C1 just in time compiler (the current default)
- auto G5 optimization
- class data sharing between JVM instances
- variety of garbage collection options
- result is about 10x faster than interpreter, but compilation does takes time

The C1 compiler:

- runtime support dynamic compilation
- object oriented optimizations
- code generation optimized for PowerPC (including AltiVec and 64 bit G5 instructions)

The Hotspot Java VM on Intel Macs will include:

- C2 just-in-time compiler
- official Java 1.3.1 support is gone
- implementing C2 on PowerPC is considered unlikely by Apple

A high quality robust implementation for Intel is ready to developer use. Java applications should work (applications with native methods need to be compiled into universal binaries), and all Java development tools are available. At this time there are known performance issues that will be fixed.

The C2 compiler:

- takes slightly more memory
- uses more advanced compilation techniques than C1 (superset of C1)
- uses the same hotspot architecture
- is run by adding `-server` to the command line (Apple is working on making it available to bundled apps and applets)
- plugs in to the JVM just like the hotspot C1 compiler
- produces better machine code which runs faster
- takes about 10x more compiler time, which may cause longer application pauses
- compilation pauses may be less on multi processor systems

Apple suggests to avoid premature optimizations, as C2 is really good at many optimizations. You should keep your code clean. Exceptions should be exceptional

C1 versus C2 on Intel based Macs using well known benchmarks suggests that C2 generally wins but not always

Network Authentication

Normally have Directory Services session, but decided not to this year since there was no major architecture changes happening between Panther and Tiger.

Open Directory (OD) provides API abstractions for read/write directory data. OD also provides authentication abstractions

- password based authentication
- challenge/response (NTLM, CRAM-MD5)
- Clear text passwords processed at the back end, not meant to send it clear.

Technology framework

- directory independent API for interacting with various directory systems
- daemon process handles client requests

Apple's position

- offers full support for legacy password methods
- aggressively pursuing Kerberos for network authentication
- add Kerberos support to majority of our software (developers should do the same)

Panther/Tiger servers employed at your customer sites with OD already setup

- OD support both password based and Kerberos based authentication
- applications should use password and or Kerberos authentication

Various frameworks available to authenticate a user

- Security.framework
- DirectoryService.framework
- PAM-based authentication

Nearly all desktop applications should use Security.framework

Server based applications should use

- DirectoryServiceFramework or PAM for passwords
- GSSAPI for Kerberos authentication and tickets

Service ACLs (Access Control Lists) part of authorization in Tiger

Kerberos

- ticket based industry standard technology for single-sign-on
- adopted by Apple, Microsoft, Novell and others
- supports packet encryption, packet signing, and mutual authentication
- GSSAPI is a common method used for network protocols
- SPNEGO and SASL use encapsulated GSSAPI tokens

Kerberos considerations

- ensure the server has matching forward and reverse DNS entries
- determine if app will run as root (may be needed in order to use keytab)
- determine if GSSAPI will satisfy requirements
- service key case vary by type (case sensitivity)

Building Automator Actions for System Administrators

<http://developer.apple.com/macosx/automator.html>

<http://developer.apple.com/documentation/AppleApplications/Conceptual/AutomatorConcepts/Articles/ShellScriptActions.html>

Automator (new for OSX Tiger) allows you to simplify repetitive tasks without programming by using Actions. Action performs a single task, and Actions can be combined into a Workflow. Building functionality from smaller focused components is a fundamental design concept of UNIX, but Automator takes that further by allowing you to assemble small tools graphically. Actions can access the core OS frameworks, the command-line environment, or special features of an application. Actions can be created with either AppleScript or Objective-C. This session focused on creating Automator Actions based on sh, perl and other scripting languages

As an aside, the Automator icon is one which gives away what Automator does, but most people miss it. If you look carefully, the Automator Robot is carrying a pipe (think Unix pipes). Automator was originally designed to graphically string together shell scripts. The ability to add AppleScript and Cocoa objects is a bonus. One of the original codenames for Automator was called Piper.

What is new and improved

- improved Run Shell Script action (available shortly)
- New AMShellScriptAction project template (Xcode 2.1) which makes it easy to create brand new actions with gui using shell script

Every Action “belongs” to an application. Shell script Actions belong to the Terminal application. In general, you define an Action’s input and output in it’s info.plist.

The Run Shell Script action belongs to Automator. You pick the shell to use in running the script, and whether input is as arguments or stdin (standard input). The action supports many scripting languages and shells. As example demo was run using Brian

Kernighan's (of UNIX fame) spell checker demo shell script of 30 years ago. A simple text file was passed to "Run Shell Script" which contained Kernighan's script. The script executes, and the results passed out as stdout (standard output). The stdout can then be processed by (or piped into) some other Action (which can be another shell script) as desired, and the results displayed graphically.

While Run Shell Script is very useful, there are limits. A obvious one is that users may introduce errors into the script if they try to change options. This is because they would have to type text modifications into the script, and presumes you know the basics of the scripting language. The workflow can be saved as an application to eliminate user error, but this makes it difficult to allow the Action to support user selected options.

Xcode 2.1 introduces Custom Shell Script Actions. These actions allow shell script-based actions to have user selectable options. Custom Shell Script Actions (introduced with Xcode 2.1) can be wrapped using Interface Builder into bundles.

In constructing workflows you need to use the "Run AppleScript" action for the purposes of having a graphical authentication dialog come up for users to authenticate a shell script. Apparently, via Automator methods, there is no direct way to run a shell script as root. If there is an official way, Apple is not saying.

Custom Shell Script Actions (using the AMShellScriptAction template in Xcode 2.1) are designed using Interface Builder, and parameters are set using Cocoa bindings. In a very, very high nutshell you follow a procedure something like this:

=====
Start in Xcode

Select "Shell Script Automator Action" and give the project a name. Note that project names cannot end in a space.

Once the project template is created, look for a few files:

First is "main.command"

- actions must have a main.command
- this is where your custom script goes
- suggest setting you path variable
- make sure you have "exit 0" at the end to exit script cleanly without locking

Second is "main.nib"

- your Interface Builder tailored user interface is stored here
- bindings for Automator is simple for the most part (versus more complex applications)
- in general, bind to Parameters (NSObjectController) with the controller key field set to "selection"

Third is "Info.plist"

- must fill in dictionary keys otherwise things will not work (AMDefaultParameters)
- Verifying the plist is recommended.

=====

Tips:

- There is no real debugger, but use "View Results" liberally
- Use "Ask for Confirmation" to pause an action
- Automator is Apple Scriptable
- You can create your own "debugging scripts". Output of stderr (standard error) is normally discarded, so you can try redirecting it somewhere.
- Custom Shell Script Actions can be saved as applications, Finder plugins, etc.

A demo was done of Custom Shell Script Action by wrapping the shell command "lsdf". The demo was for the scenario in which you want to eject a disk, and get the error message that it cannot be done because the disk is in use (but it does not say why it is in use). The "lsdf" command output, if you can read it, might give you the answer.

So the workflow is to take the identify the disk and the Finder error message, run lsdf to figure out what the problem is, and display the result to the user (e.g. the culprit is the application called "foo"). By saving the action as a Finder plugin with a descriptive name (e.g. "Why can I not eject this disk?"), all the user has to do is select the action from the Contextual menu associated with the drive in the Finder.

Now as to why this function is not already part of the Finder is another question....

10th Annual Apple Design Awards 2005

For each category we list the winner first, followed by the runner-up.

Best Mac OS X Student Product

GraphClick 2.5 by Simon Bovet (www.arizona-software.ch)

iMap 3.1 by Peter Schols (www.biovolution.com)

Best use of Open Source

Osirix 1.6.4 (homepage.mac.com/rossetantoine/osirix/Index2.html)

Blender 2.36 by Blender Foundation (www.blender3d.org)

Best Mac OS X Server Solution

Quicksilver InfiniBand Software for Mac OS X 3.1 by SilverStorm Technologies
(www.infinicon.com)

Elektron 1.0.1 by Corriente Networks LLC (www.corriente.net)

Best OS X Scientific computing solution

DataTank 2005-5 by Visual Data Tools, Inc (www.visualdatatools.com)

Osirix 1.6.4 (homepage.mac.com/rossetantoine/osirix/Index2.html)

Best Mac OS X Entertainment Product

World of Warcraft 1.3.1 by Blizzard Entertainment (www.blizzard.com)

Jammin' Racer 1.01 by DanLabGames (www.danlabgames.com)

Best Mac OS X User Experience

Delicious Library 1.5 by Delicious Monster Software (www.delicious-monster.com)

3d Weather Globe and Atlas Mac OS X Edition by The Software Mackiev Company
(www.mackiev.com)

Best Mac OS X New Product

ComicLife 1.1 by Plasq (www.plasq.com)

Delicious Library 1.5 by Delicious Monster Software (www.delicious-monster.com)

Best use of Mac OS X Tiger Technology

Transmit 3.2 by Panic, Inc (www.panic.com/transmit)

iSale 1.7 by Equinux USA, Inc (www.danlabgames.com)

Summary

As always, the notes above represent only a sampling of the sessions that were available. The conference had both developer and enterprise IT tracks.

By the end of the conference (or very shortly thereafter) at least two other major development houses got their software to compile, build and launch under OS X Tiger for Intel Preview. Blizzard got World of Warcraft to run, and Epic got the engine for the Unreal Tournament 2004 built. World of Warcraft for OS X for PowerPC is a Carbon application using MetroWorks Codewarrior in conjunction with the Xcode performance tools, while the Intel build is done using Xcode. While these software houses are game developers, the application source code is not trivial. This is not to say they can ship it today. There is tuning and optimizations to do, and newly found bugs to squash. One advantage of building for either another architecture and/or another OS platform is that it helps to find obscure bugs (or obvious bugs that should have been found earlier). So while the developer effort required to make an application universal depends on the application, things are off to a good start.

<http://www.insidemacgames.com/news/story.php?ArticleID=11430>

<http://icculus.org/cgi-bin/finger/finger.pl?user=icculus&date=2005-06-10&time=21-42-20>