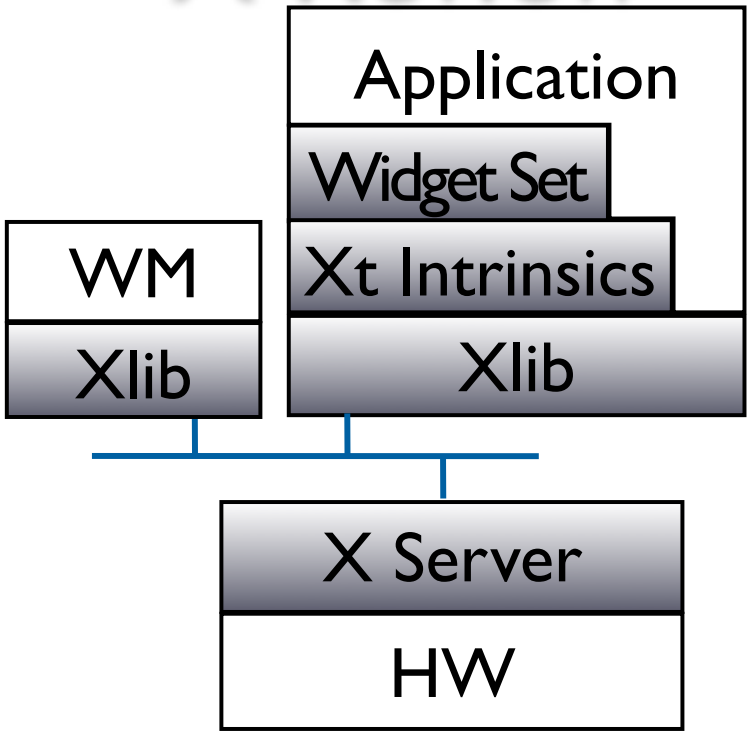
X Review

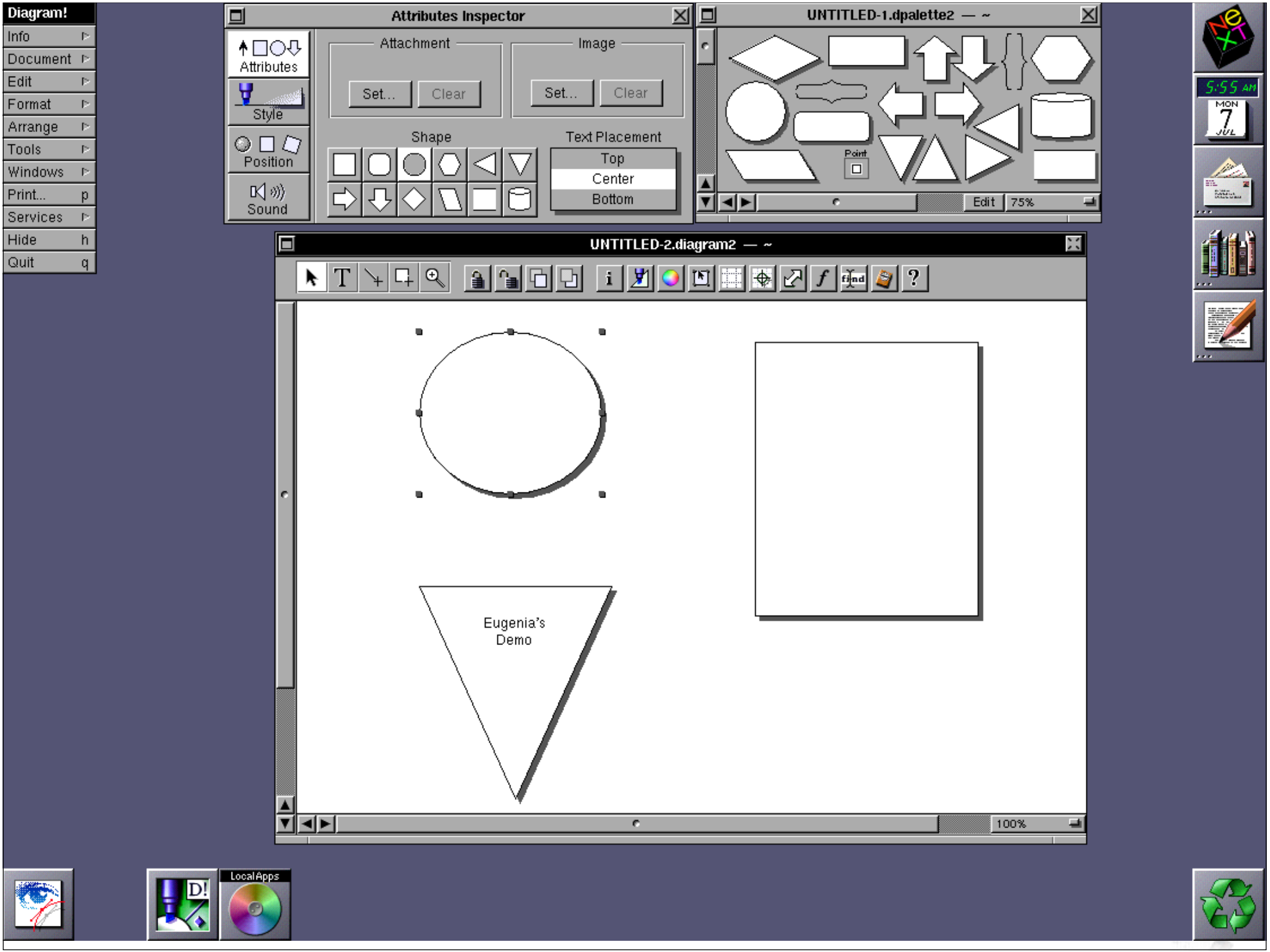


- Distributed window system
  - Server is the user's Terminal
  - Client runs the application
- Highly modular
  - (exchange WM, Widget Set)
- Available for virtually every OS
- Very successful in the UNIX world (with Motif)
- Many Open Source implementations available
- Xt intrinsics already allow OO GUI application programming


Mac OS X Snow Leopard
Mac OS X Snow Leopard
The world's most advanced operating system. Finely tuned.
Version 10.6
Apple

Mac OS X Roots: NeXT

- 1985 Apple kicks out Jobs, who founds NeXT
- 1988 NeXTSTEP 0.8: 68k, Mach kernel, 4.3BSD, Display PostScript, new Objective-C, platform for first web browser at CERN
- 1990 NeXTSTEP 2.0: Interface Builder (OO)
- 1993 Intel, SPARC, HP PA-RISC support (fat binaries), NeXT hardware dead
- NeXT&Sun: OPENSTEP, used by SunOS, HP-UX, MacOS, also on Windows NT, GNUstep


Diagram!
Info
Document
Edit
Format
Arrange
Tools
Windows
Print... p
Services
Hide h
Quit q
Attributes Inspector
Attributes
Style
Position
Sound
Attachment
Set...
Clear
Image
Set...
Clear
Shape
Text Placement
Top
Center
Bottom
UNTITLED-1.dpalette2 — ~
Point
Edit
75%
UNTITLED-2.diagram2 — ~
Eugenia's
Demo
100%
LocalApps

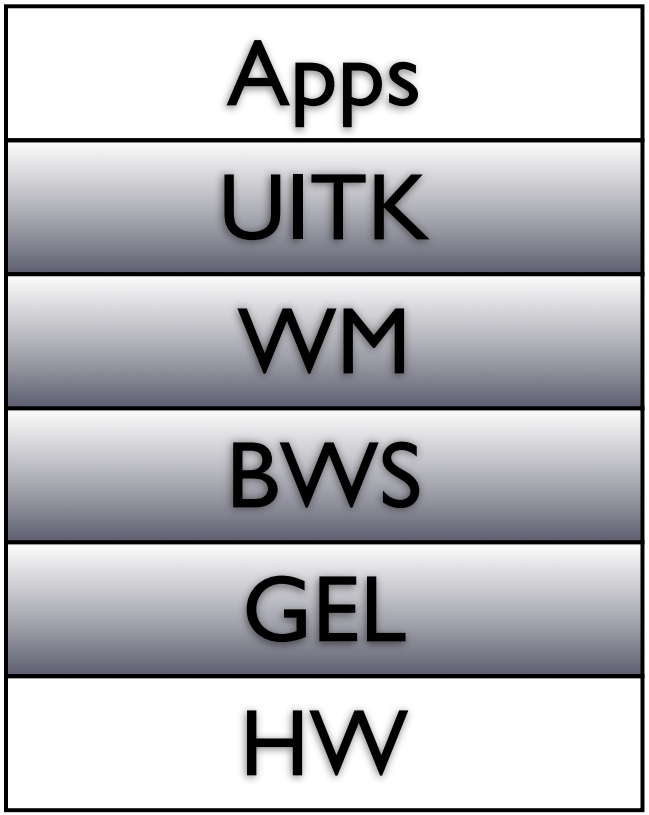
Mac OS X Roots: NeXT

- 1996 Apple purchases NeXT, Jobs back at Apple, begins replacing Mac OS towards NeXTSTEP
- 1999 Apple releases Mac OS X Preview

➡ Mac OS X is directly derived from NeXTSTEP/ OPENSTEP (Mach/BSD, Objective-C, Cocoa, Interface Builder, NetInfo), and combines this with other Apple technologies (UI, Multimedia)

It's spelled "Ten"

Mac OS X: Architecture



Cocoa, Carbon, Java

WindowServer (user-level process)

Quartz 2D / Core Graphics, IOKit

Base Operating System

- Darwin (Open Source for PowerPC+Intel)
- Based on a Mach microkernel + BSD Unix
- Protected memory, preemptive multitasking
  - Single application cannot corrupt/freeze entire system

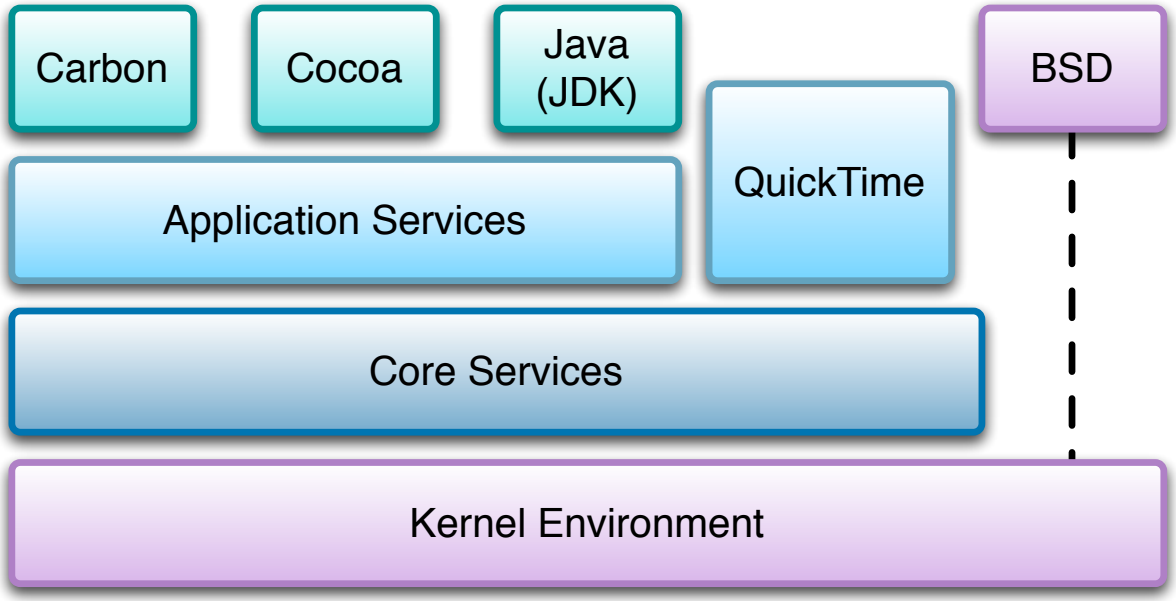
Graphics and Event Library

- Quartz 2D / Core Graphics, Display PDF
- Roots: NeWS, NeXTSTEP (Display PostScript)
- Vector-based, resolution-independent
- Quartz Extreme: GPU-based compositing (pipe entire screen through OpenGL if available)
- Quartz Extreme 2D / QuartzGL: GPU-based UI rendering (still experimental)
- Vista's XPS is similar to (Display) PDF
- The Event Library is contained in IO Kit

Demo: Quartz Extreme

User Interface Toolkit

- Cocoa AppKit, OO framework
- Implements Aqua Look&Feel
- Other part of Cocoa (Foundation framework) contains threads, sockets, etc.
- Written in Objective-C



- Cocoa is the recommended API for Mac OS X
  - Can be used with Objective-C or Java*
- Java SE 6 SDK is used for 100% Java/Swing applications
- Carbon is an updated version of the old Macintosh Toolbox
  - Used to easily port existing applications
  - Carbon apps run on Mac OS 9 and Mac OS X
- BSD is used for existing standard Unix applications
- WebObjects for web-based services

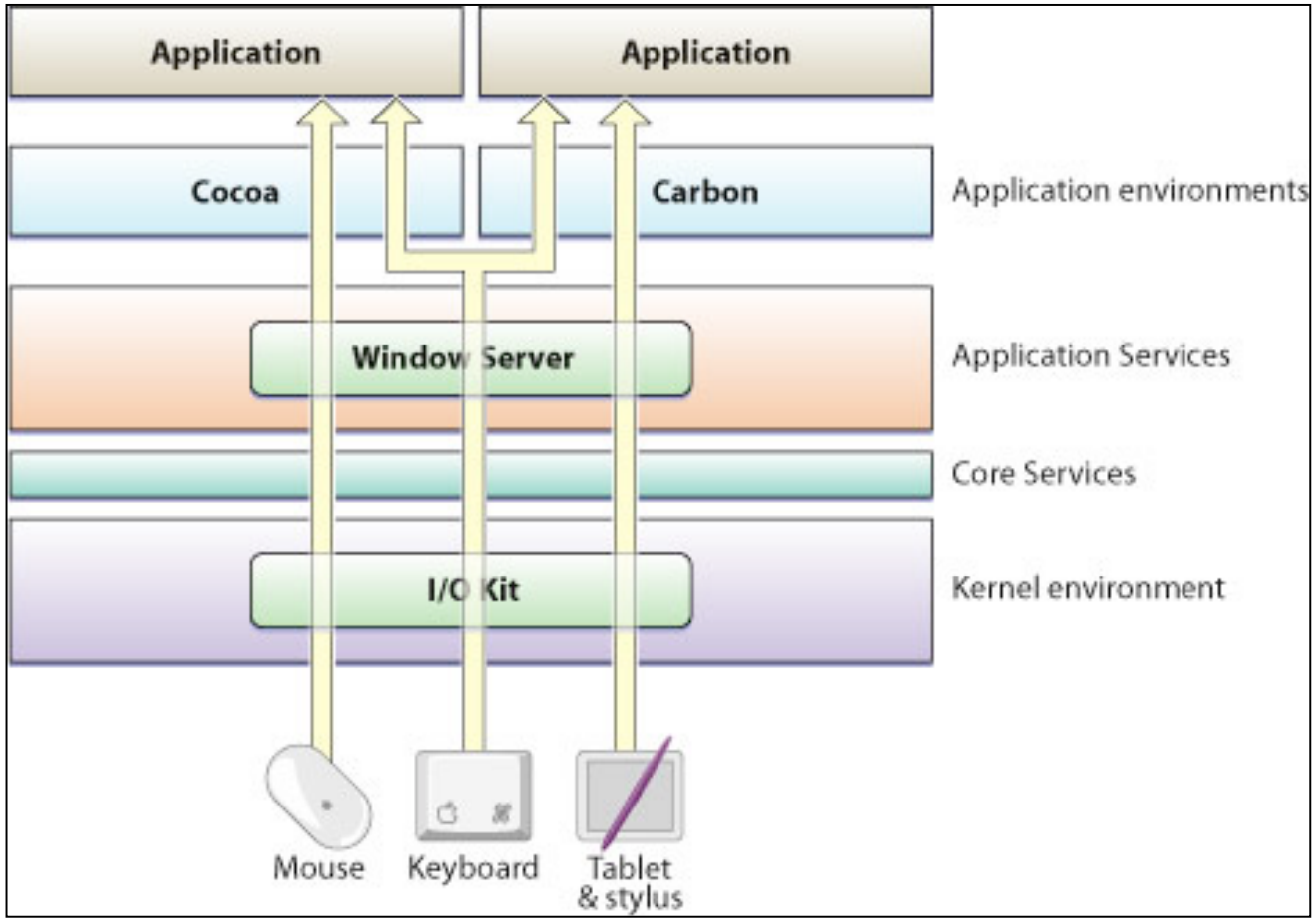
* Multi-API Problems

- Mac OS X has tried to please everyone
- Problem: High cost for supporting many APIs
- Consequences in 2006:
  - No new Java bindings for new Cocoa features in Mac OS X 10.5 and beyond (= "Use Objective-C!")
  - New Intel Macs do not support Classic Mac OS emulation anymore (= "Classic's dead!")
- But: New bridges for Ruby and Python since Leopard

Steve Jobs burying Mac OS 9 at WWDC 2002 (personal photo)

Event Handling

- Similar to our Reference Model
- Window Server distributes events to per-application (really per-process) queues



Objective-C

- Implementation language of the Cocoa framework
- Created in 1983 to combine OO principles with C
  - Inspired by Smalltalk
- In its concepts very similar to Java, unlike C++
- Dynamic typing, binding, and loading
- Protocols are analogous to Java's interfaces
- Classes are objects of type Class
  - Reflection possible (not in C++)
- Introduces few new language constructs to C

Objective-C: New C Language Constructs

- `status = [socket connectToAddress:127.0.0.1 withPort:80]` is analogous to Java's
  `status = socket.connectToAddressWithPort(127.0.0.1, 80)`
- `–` for instance methods, `+` for class methods
- `id` corresponds to Object, `self` corresponds to this
- `@` compiler directives
  - `@interface..@end, @implementation..@end,…`
- Interface categories allow transparent subclassing

Dynamic Typing/Binding/Loading

- C++ is a static language, Java and Obj-C are dynamic
- Calling known methods of a subclass
  - Example: Building objects from data read from a file
  - In C++, you cannot directly call a method defined in a subclass
    - You would have to do a dynamic_cast<>
  - Obj-C (and Java) move this check to run-time
- Fragile Base Class Problem
  - To add a method to a superclass, all subclasses must be recompiled
  - In C++, a superclass must contain all methods any subclass will use
  - Dynamic Binding avoids bloated superclasses, and minimizes recompilation due to this problem

Demo: Dynamic Binding

Objective-C 2.0 (since Leopard)

- Supports automatic accessor method generation for object properties
- Allows Java-like short-hand syntax to access properties
  - `[object setName:newName];`
    `object.name = newName;`
- Features an optional garbage collection
  - only works for Objective-C Objects
  - you still have to manage any C pointers yourself
- Supports fast enumeration of collections
  - `for (object in list) [object doSomething];`

Manual Memory Management

- Reference counting
  - Each object stores how often it is referenced by other objects
  - If it is not used anymore, call dealloc (free memory)
- Only five methods affect reference counts
  - alloc—allocate object and set reference count to 1
  - copy—copy an object. Copy will have reference count 1
  - retain—increment reference count
  - release—decrement reference count
  - autorelease—automatically decrement reference count sometime after the current method

Cocoa

- The UITK (and general toolkit) of Mac OS X
  - Evolved out of NeXTSTEP
  - Cocoa's class names still give away its heritage (NSObject,…)
- Two main parts
  - Foundation: Basic programming support
    - NSObject, values, strings, collections, OS services, notifications, archiving, Objective-C language services, scripting, distributed objects
  - AppKit: Aqua interface
    - Interface, fonts, graphics, color, documents, printing, OS support, international support, InterfaceBuilder support
    - Largest part (over 100) are interface widgets

Cocoa Class Hierarchy

NSObject
- NSEvent
- NSResponder
  - NSWindow
  - NSView
    - NSControl
      - NSButton etc.
  - NSApplication
- NSCell (lightweight controls)
- NSMenu
- NSMenuItem
- etc.

- Fairly flat hierarchy
- Reason: Delegates, and categories can be used to mix in functionality
  → no deep subclassing required
- Full hierarchy:
  see Online Help in Xcode

Delegates

- Similar to Listeners in Java
- A delegate is a class whose methods are called from another class that wants to plan for extending its functionality
  - E.g., you can write a delegate class for NSTableView that will get called when the user changes the width of a column
- Delegates solve the fragile base class problem
  - If NSTableView is changed, the delegate class doesn't need to be recompiled

Example: NSTableView Delegate Methods

Moving and resizing columns

– tableView:didDragTableColumn:

– tableViewColumnDidMove:

– tableViewColumnDidResize:

Selecting in table

– selectionShouldChangeInTableView:

– tableView:shouldSelectRow:

– tableView:shouldSelectTableColumn:

– tableViewSelectionIsChanging:

– tableViewSelectionDidChange:

Responding to mouse events

– tableView:didClickTableColumn:

– tableView:
mouseDownInHeaderOfTableColumn:

Editing a cell

– tableView:
shouldEditTableColumn:row:

Displaying a cell

– tableView:
willDisplayCell:forTableColumn:row:

Displaying tooltips

– tableView:
toolTipForCell:rect:tableColumn:
row:mouseLocation:

Allowing variable height rows

– tableView:heightOfRow:

Protocols

- Similar to Java's interfaces
- Mostly replace the need for multiple inheritance

Categories: Extending a class without subclassing

- Example: Add a "find&replace" method to NSString
  - Could subclass "MyNSString"—but then have to change all code to use that new class
  - Could change NSString itself—but that would require access to the source code for that class (and recompiling, rebinding any code using it,…)
  - Instead: Create a category "NSStringExtensions":

```
@interface NSString (NSStringExtensions)
- (NSString *)find:(NSString *)findString
          replaceWith:(NSString *)replaceString;
@end
```

- Any code using NSString can now use the find:replaceWith: method
- An advantage of the Dynamic Loading possible with Objective-C

Demo: Categories

Events and Actions

- Events are generated by user or system (e.g., periodic events)
  - Corresponds to input "I" from our formal widget model
- Actions are generated by objects (e.g., menus) in response to lower-level events
  - Corresponds to actions "A" from our formal widget model
  - InterfaceBuilder lets developer connect actions to custom objects (e.g., from a button to a custom class), using "IBAction" constant in the source

Responders

- Most UI objects are subclasses of NSResponders and can respond to events
- NSApplication can find a responder that can handle an event (respondsToSelector), then call its method directly
- Framework takes care of Responder Chain
  - Events are passed on along the responder chain (key window → main window → application) until they can be handled by some object
- Applications, windows, views, etc. can be extended by adding a delegate without having to subclass them

Outlets

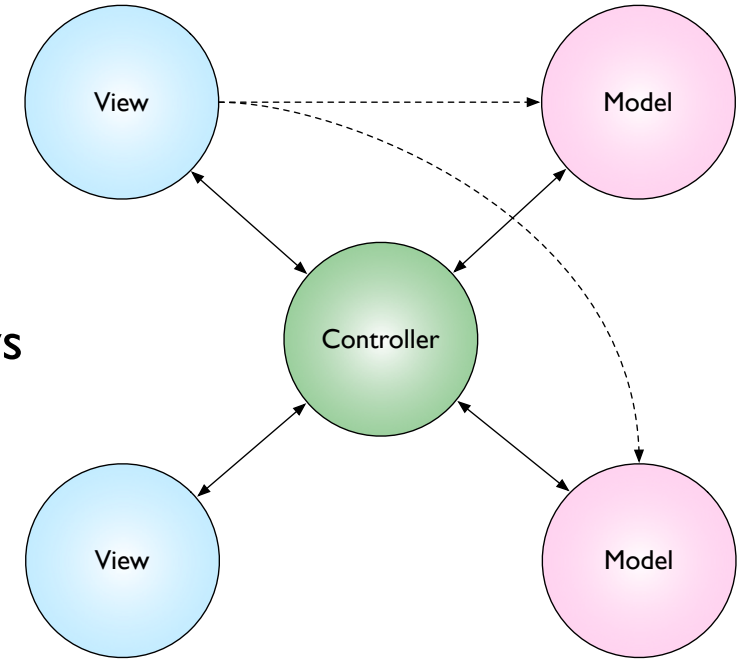
- Outlets are instance variables to refer to other objects
  - InterfaceBuilder knows about them and lets the developer connect outlets graphically ("IBOutlet" constant)
  - Example: A custom class that wants to display a result in a text field needs an outlet

Interface Builder

- Graphical tool to create UIs for Cocoa (and Carbon) applications
- Tied into development environment (Xcode)
- Resources are stored in xib files (XML Interface Builder)
  - used to be nib (NeXTSTEP Interface Builder)

- An application reads its main xib file automatically when it starts up
- Additional xib files can be read when needed (for transient dialogs, etc.)

Aside: MVC



- Model
  - store and change data
  - should never know about specific views
- View
  - parts of the UI
  - sometimes have a reference to models
- Controllers
  - sets up connections
  - translate actions into model specific commands (click ➔ add item)
  - relays changes in the model to appropriate views (disable a button, update a table)

Demo:
Interface Builder Basics

Interface Builder: Defining the Look

- Interface Builder supports the developer when visually laying out the user interface
- Guides (similar to Photoshop) help to adhere to the visual aspects of the Aqua Design Guidelines (the Mac OS X Look and Feel)
- This helps design the View in the MVC (Model–View–Controller) paradigm

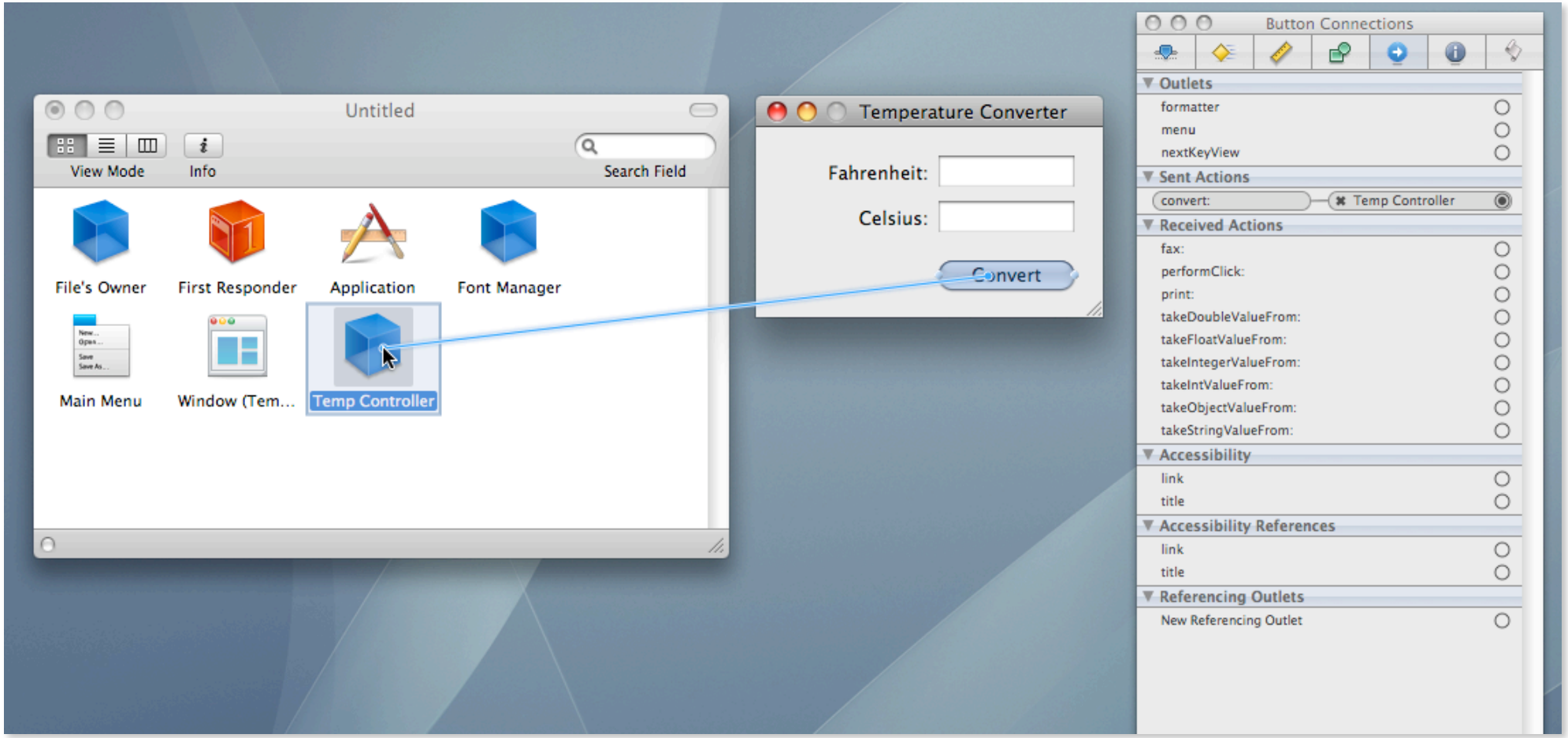
Demo: Interface Builder —Layout Support

Interface Builder: Defining the *Feel*

- Not just visually define the widgets in a UI (specify static UI layout—which is what most UIDS support), but also connections between widgets and custom classes of the application (dynamic UI behavior)
- Linking View and Controller
- Can even synthesize Controller to directly link View and Model
- UI can be tested inside Interface Builder without compiling or writing any code

Example

User input via an NSButton is connected to the convert() method of a custom TempController class in an application.



Demo: Interface Builder Connecting Actions, Testing

Interface Builder: Defining the Feel

- Suggests a more user-centered implementation process that starts with the user interface, not the application functionality
  - Interface Builder synchronizes with source code skeleton that can then be filled in
  - Interface Builder uses special constants to include hints about outlets and actions in the source code

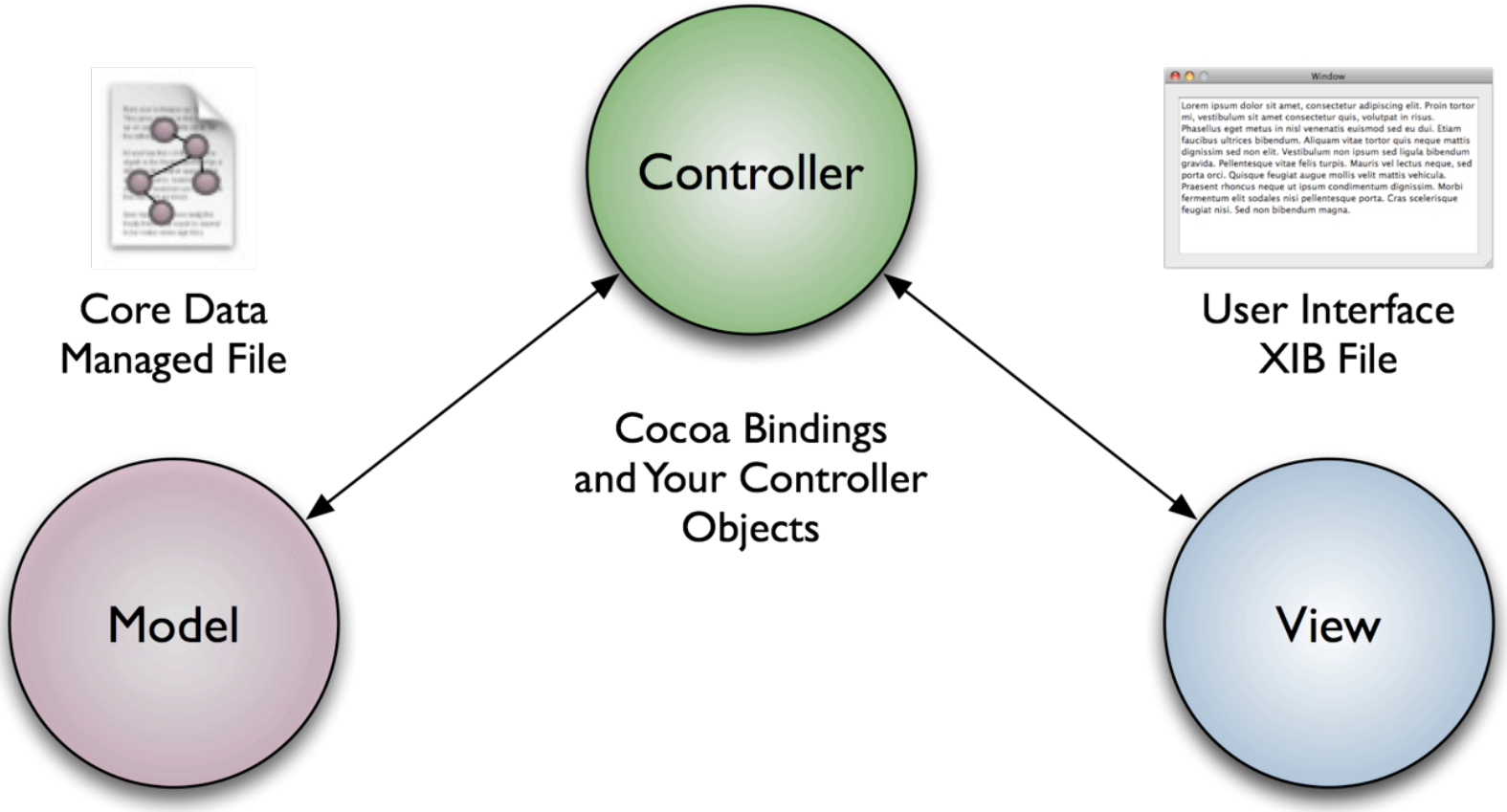
Cocoa Bindings

- Cocoa Bindings are an approach to keep MVC Model and MVC View synchronized without having to write a lot of glue code
- Example: Keeping a displayed table of students (View) synchronized with the corresponding array (Model) of data in memory, and also with a label showing the number of students (another View)
- Cocoa Bindings define the MVC *Controller*

Demo: Cocoa Bindings in Interface Builder

Core Data: Defining the MVC Model

- Object-graph management and persistence framework
- To define application data model graphically
- Provides common functionality
  - Undo, Redo
  - Persistence (save to disk, read from disk in XML or SQLite format)


Controller
Core Data
Managed File
Window
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin tortor mi, vestibulum sit amet consectetur quis, volutpat in risus. Phasellus eget metus in nisl venenatis euismod sed eu dui. Etiam faucibus ultrices bibendum. Aliquam vitae tortor quis neque mattis dignissim sed non elit. Vestibulum non ipsum sed ligula bibendum gravida. Pellentesque vitae felis turpis. Mauris vel lectus neque, sed porta orci. Quisque feugiat augue mollis velit mattis vehicula. Praesent rhoncus neque ut ipsum condimentum dignissim. Morbi fermentum elit sodales nisi pellentesque porta. Cras scelerisque feugiat nisi. Sed non bibendum magna.
User Interface
XIB File
Cocoa Bindings
and Your Controller
Objects
Model
View

Demo: Core Data

Core Image, Core Audio, Core Animation

- Core Image: Framework for GPU-accelerated image display, contains a large and extensible set of Image Units (visual effects, transitions, and filters) that can be combined at runtime, and used by any app developer
- Core Audio: toolbox of Audio Units for audio effects and digital audio signal processing
- Core Animation: implicit animation support for the UI

Demo:
Core Image Fun House

Demo: Core Audio—AULab

Demo: Core Animation

Webclips, Dashcode, Automator

- Webclips
  - Windows to websites
  - Updated automatically
- Dashcode
  - Programming environment for Dashboard Widgets
  - Bases on web technology (JavaScript, HTML, Images)
- Automator
  - End user scripting tool
  - Like graphical shell scripts

Demo: Webclips, Dashcode, Automator

More Information

- Apple Developer Documentation: http://developer.apple.com/