

You Have an App for That: How to Build an iPhone App



Ellis Holman
IBM Corp.

Wednesday, August 10, 2011
Session Number 9774

Disclaimer



THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

Other company, product, or service names may be trademarks or service marks of others.

Things to consider before jumping into code

- How does a potential user of the app view it ?
- How does the app relate to different cultures?
- What are the limitations of any potential user(s)?
- What are the limitations of the display presentation?
- Will the app be highly interactive?
- Does your app solve a unique problem?



© 2010 CBS Interactive

Basic Physical Ergonomics

- Here are a couple of the most important physical-, cognitive- and ergonomic-related truths about the iPhone
 - Fingers are not mouse pointers
 - Finger surface area is not equal to one pixel
 - Tappable objects that are way too small, make the interface frustrating to use
 - People will not continue to use an app that frustrates them – SURPRISE!

Designing the UI with templates to speed the process

- Download the iPhone GUI Photoshop template or the iPhone PSD Vector Kit
 - Both are collections of iPhone GUI elements that will save a lot of time in getting started
 - If the layout has been solidified during sketching, drawing up the screens will be less of a layout exercise and more about the actual design of the app

Thoughts on ergonomic challenges

- Here are some ways to solve these ergonomic challenges:
 - Make buttons and other tappable objects bigger
 - If making a button bigger is impossible, then enlarge the clickable area to be bigger than the button itself
 - Reduce the number of options on each screen, and make the selection process sequential
 - Implement multi-touch gestures within the interface

Think about substance over 'flash', simplicity over complication

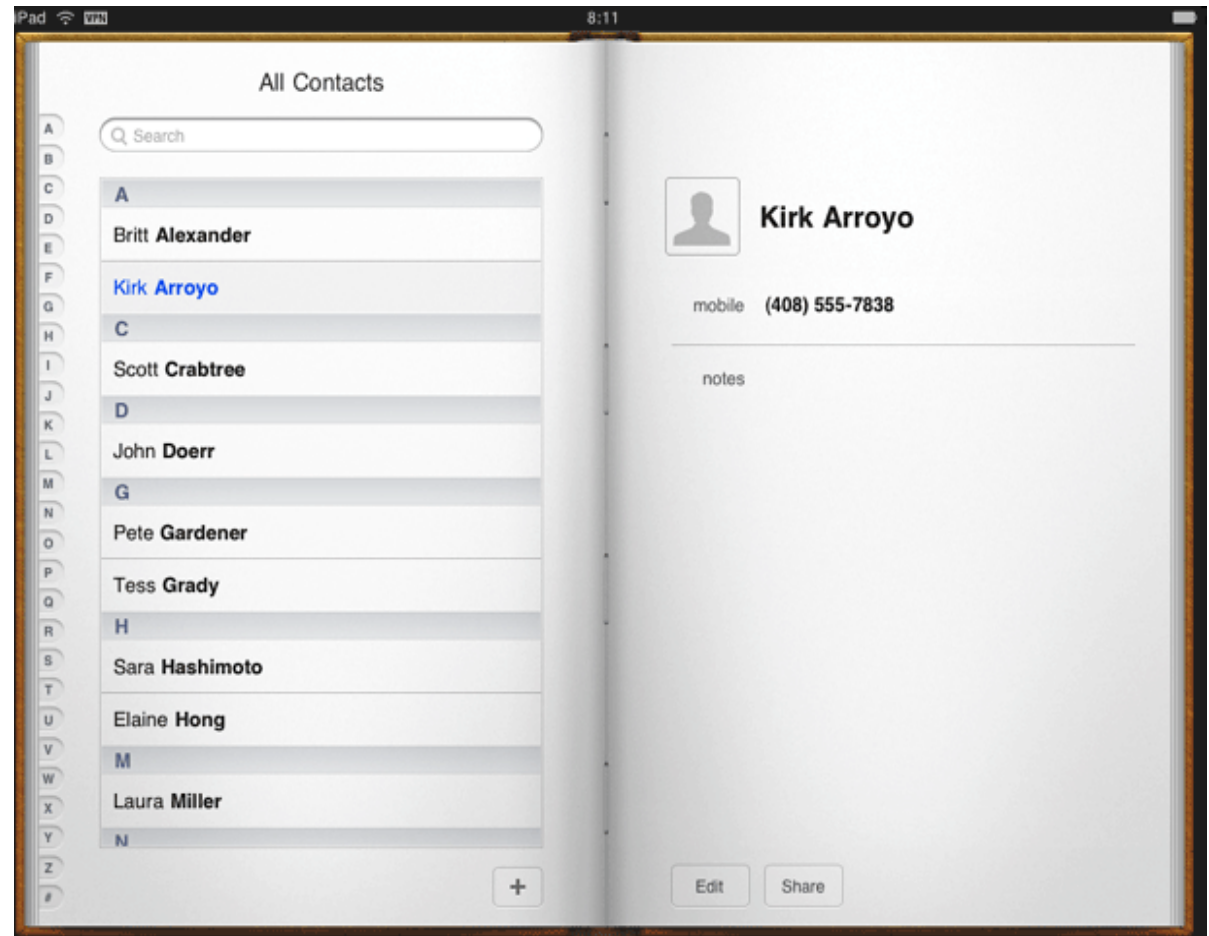


- 7 Based on which design works better overall, the one on the right wins. The one on the left looks great when checking it out on a iPhone but performs poorly in a real-world context. Simple is good

The more an app resembles a 'real' item, the easier it will be for a user

Most everyone
knows what an
address book is and
how to use it

If the app mimics the
look and feel of a real
one, people will find it
both easy and
comfortable to work
with



Basic design for building iPhone apps

- At a high level, the process for creating an iPhone application is similar to that for creating a Mac OS X application
- Both use the same tools and many of the same basic libraries
- Smaller size of the iPhone screen means that an application's user interface should be well organized
 - Always focus on the information the user needs most



The display is at the heart of the user's experience

- People view beautiful text, graphics, and media on the display
- They also physically interact with the Multi-Touch screen to drive their experience (even when they can't see the screen)
- Although displays of different dimensions and resolutions can have different effects on user experience with an app, some effects apply to all iOS-based devices:
 - The comfortable minimum size of tappable UI elements is 44 x 44 points.
 - The quality of app artwork is very apparent
 - The user's focus is on the content

Device	Portrait	Landscape
iPhone 4	640 x 960 pixels	960 x 640 pixels

Other language support for the iPhone

- A JAVA J2ME stack has been demonstrated to run on an iPhone, though it involved jailbreaking
- It is not permissible to install a .NET Framework or similar runtime on an iPhone
 - Using Novell's commercial MonoTouch framework it is possible to achieve similar results
 - MonoTouch uses a custom fork of the Mono Project to compile all CLI bytecode in .NET to native ARM machine-code ahead of time
- iOS does not support Adobe Flash.
 - Flash movies on web pages cannot be viewed in Mobile Safari

Tools to help you get started building an app

- Join the Apple iPhone Developer Program (\$99)
 - Gives you access to software development kit (SDK)
- Get an iPhone
- Get an Intel-based Mac computer with Mac OS X 10.5.5 or higher
 - The SDK runs on this, NOT Windows
- Prepare a Non-Disclosure Agreement
- Download and install the latest version of the iPhone SDK if you don't already have it
- A spiral bound notebook to keep notes in and do display sketches

Apple's software development kit provides the software 'tools' to build an app

Apple provides a software development kit for \$99.00

The SDK contains:

- Cocoa Touch which provides an abstraction layer of iOS, for the iPhone written in Objective-C language
 - Multi-touch events and controls
 - Accelerometer support
 - View hierarchy
 - Localization (i18n)
 - Camera support
- Media
 - OpenAL
 - audio mixing and recording
 - Video playback
 - Image file formats
 - Quartz
 - Core Animation
 - OpenGL ES
- Core Services
 - Networking
 - Embedded SQLite database
 - Core Location
 - Threads
 - CoreMotion
- Mac OS X Kernel
 - TCP/IP
 - Sockets
 - Power management
 - File system
 - Security

Included with the SDK is an iPhone simulator

- The iPhone Simulator is a program used to emulate the look and feel of the iPhone on the developer's desktop
- Originally called the Aspen Simulator, it was renamed with the Beta 2 release of the SDK
- The iPhone Simulator is not an emulator and runs code generated for an x86 target
- This will give a developer how an app will look without having to get the app loaded onto an iPhone

Typical behaviors of an iPhone application

Typical behaviors include:

- Initializing the application
- Displaying a window
- Drawing custom content
- Handling touch events
- Performing animations

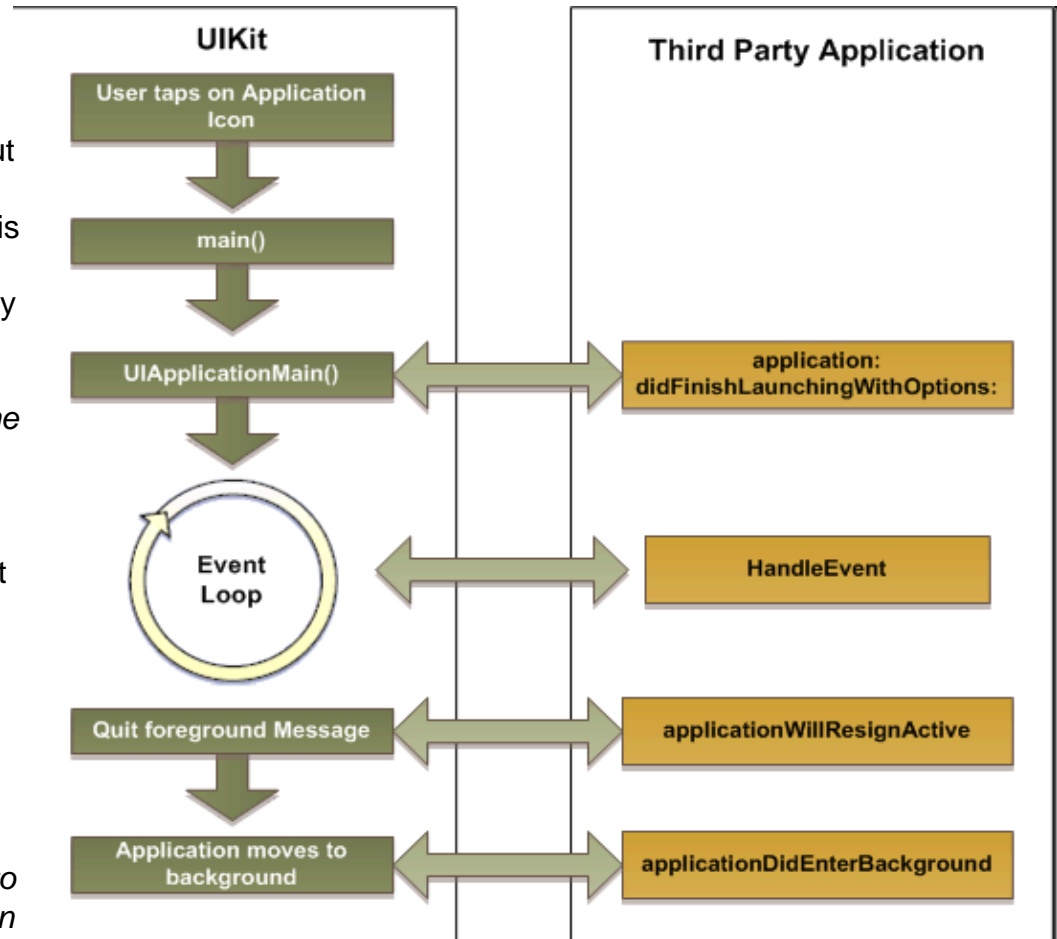
The basis for iPhone apps is Objective-C

- It is a simple computer language designed to enable sophisticated object-oriented programming.
- Objective-C is defined as a set of extensions to the standard ANSI C language.
- Additions to C are mostly based on Smalltalk, one of the first object-oriented programming languages.
- Objective-C consists of several parts:
 - An object-oriented programming language
 - A library of objects
 - A suite of development tools
 - A runtime environment

Application phases for iOS V4 with suspension

Because of background applications support in **iOS 4**, following are the possible state of any third party application:

- **Not running**: The application has not been launched
- **Inactive**: The application is running in background but not receiving events
- **Active**: The application is running in foreground and is receiving events
- **Background**: Most applications enter this state briefly on their way to being suspended
 - *An application that requests extra execution time may remain in this state for a period of time*
 - *In addition, an application being launched directly into the background enters this state instead of the inactive state*
- **Suspended**: The application is in the background but is not executing code
 - *The system moves application to this state automatically and at appropriate times*
 - *While suspended, an application is essentially freeze-dried in its current state and does not execute any code*
 - *During low-memory conditions, the system purges suspended applications without notice to make more space for the foreground application*



Fundamental application flow for the iPhone

Main patterns that you'll want to consider are:

- Delegation
 - a pattern in which one object sends messages to another object specified as its delegate to ask for input or to notify the delegate that an event is occurring.
- Model
 - Represent data such as SpaceShips and Weapons in a game, ToDo items and Contacts in a productivity application, or Circles and Squares in a drawing application
- View
 - Objects that know how to display data (model objects) and may allow the user to edit the data
- Controller
 - Objects that mediate between models and views
- Target-Action
 - Mechanism enables a view object that presents a control
 - An object such as a button or slider—in response to a user event
 - Send a message (the action) to another object (the target)

Building the user interface

- Next Interface Builder has been around awhile and is used to design user interfaces
- It allows developers/designers to layout their user interfaces in a drag-and-drop visual format instead of having to construct the whole interface using code
- This can greatly speed up the development process, and, it can make it possible for non-developers who are designers to be able to do their work.
- Its been used to design window layouts for Mac applications for quite some time.
- Apple engineers have now made it possible to design iPhone GUIs on it as well!



One of the most important architectural details is to define the application delegate object

- The application delegate object works in tandem with the standard UIApplication object to respond to changing conditions. The application object does most of the heavy lifting, but the delegate is responsible for behaviors, including the following:
 - Setting up the application's window and initial user interface
 - Performing any additional initialization tasks needed for your custom data engine
 - Opening content associated with the application's URL schemes
 - Responding to changes in the orientation of the device
 - Handling low-memory warnings
 - Handling system requests to quit the application
- At launch time, the most immediate concern for the delegate object is to set up and present the application window to the user
- The delegate should also perform any tasks needed to prepare an application for immediate use, such as restoring the application to a previous state or creating any required objects
- When the application quits, the delegate needs to perform an orderly shutdown of the application and save any state information needed for the next launch cycle

Using blocks to segment units of work

- Blocks are a segment of code that can be executed at any time
- Essentially portable and anonymous functions that one can pass in as arguments of methods and functions or that can be returned from methods and function
- A block may also be assigned to a variable and then call it just as you would a function

```
int (^Multiply)(int, int) = ^(int num1, int num2) {  
    return num1 * num2;  
};
```

```
int result = Multiply(7, 4); // result is 28
```

Dealing with strings

- Strings embedded in application code, must be extracted, localized, and then reinserted back into the code.
 - To simplify this process—and to make the maintenance of application code easier—Mac OS X and iOS provides the infrastructure needed to separate strings from the code and place them into resource files where they can be localized easily
- Resource files that contain localizable strings are referred to as **strings** files because of their filename extension, which is .strings
- When you need to display a string, pass the string to one of the available string-loading routines
- What returned is the matching value string containing the text translation that is most appropriate for the current user

A Word About Memory Management

- iOS is primarily an object-oriented system, so most of the memory you allocate is in the form of Objective-C objects
- iOS uses a reference counting scheme to know when it is safe to free up the memory occupied by an object
- When an object is first created, it starts off with a reference count of 1
- Clients receiving that object can opt to retain it, thereby incrementing its reference count by 1
- If a client retains an object, the client must also release that object when it is no longer needed
- Releasing an object decrements its reference count by 1
- When an object's reference count equals 0, the system automatically reclaims the memory for the object
- There is no garbage collection mechanism, as there is for MAC O/S v10.5 and later

Supporting multiple orientations on the iPhone

- iPhone-only applications may only have one launch image
 - PNG format and measure 320 x 480 pixels.
 - Launch image file named Default.png.
- For iPhone 4 high resolution, optionally include an additional launch image
 - PNG format and measure 640 x 960 pixels.
 - Name it Default@2x.png
 - This image will get picked up by the iOS if your app is running on an iPhone 4
- Apps not running on an iPhone 4: if both Default.png and Default@2x.png are provided, iOS will automatically pick up Default.png as the launch image.

Starting the application up

- Every application is responsible for creating a window that spans the entire screen and for filling that window with content.
- Graphical applications running in iOS do not run side-by-side with other applications
- Windows provide the drawing surface for your user interface, but view objects provide the actual content.
 - A view object is an instance of the UIView class that draws some content and responds to interactions with that content.
 - iOS defines standard views to represent things such as tables, buttons, text fields, and other types of interactive controls.
 - Any of these views can be added to a window, or custom views can be defined
 - Use subclassing UIView and implementing some custom drawing and event-handling code
- At launch time, the goal is to create the application window and display some initial content as quickly as possible.
- The window is unarchived from the MainWindow.xib nib file.
- When the application reaches a state where it is launched and ready to start processing events, the UIApplication object sends the delegate an `applicationDidFinishLaunching:` message
 - This message is the delegate's cue to put content in its window and perform any other initialization the application might require

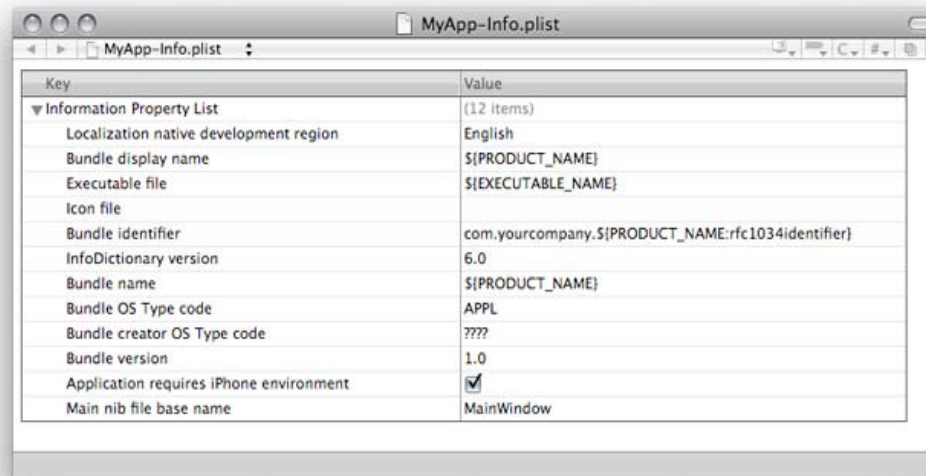
Dealing with gestures

- There may be multiple fingers touching the device at one time
- Use those events to identify complex user gestures
- The system provides help in detecting common gestures such as swipes,
- The application is responsible for detecting more complex gestures.
- When the event system generates a new touch event, it includes information about the current state of each finger that is either touching or was just removed from the surface of the device.
- Each event object contains information about all active touches,
 - In this way actions of each finger can be monitored with the arrival of each new event.
 - Movements of each finger can be tracked from event to to detect gestures, which you can apply to the contents of your application.



The property list allows an app to retain state information

- Property-list files are XML files that organize data into named values and lists of values using simple data types
- These data types allow creation, transportation, and storage of structured data in an accessible and efficient way
- **Information-property list** file commonly referred to as info-plist files, contain essential information used by your application and iOS.
- **Entitlements** file define properties that provide an application access to iOS features (such as push notifications) and secure data (such as the user's keychain)



Key	Value
Information Property List	(12 items)
Localization native development region	English
Bundle display name	\$(PRODUCT_NAME)
Executable file	\$(EXECUTABLE_NAME)
Icon file	
Bundle identifier	com.yourcompany.\$(PRODUCT_NAME)rfc1034identifier
InfoDictionary version	6.0
Bundle name	\$(PRODUCT_NAME)
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone environment	<input checked="" type="checkbox"/>
Main nib file base name	MainWindow

XCode Editor

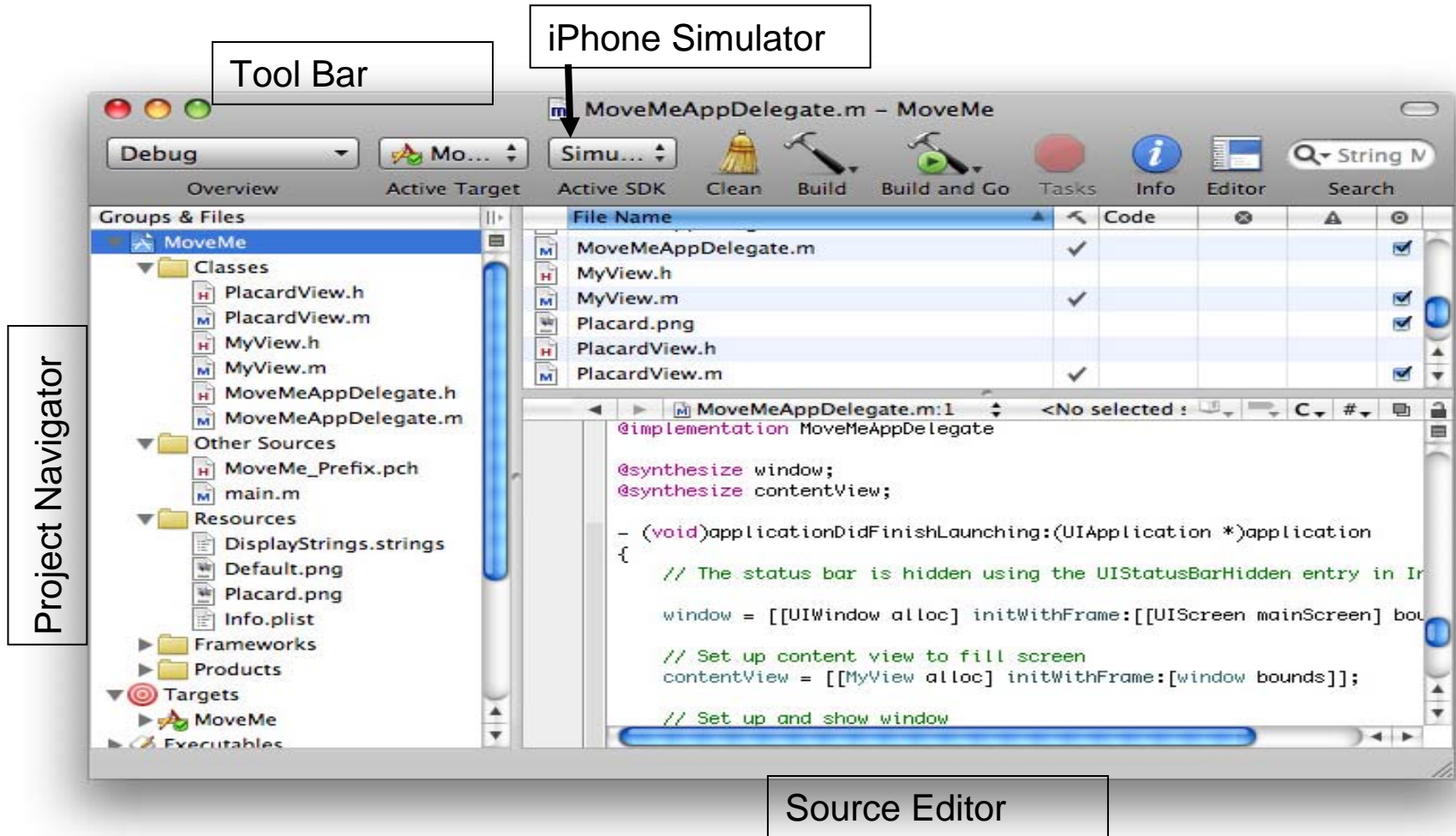
Using blocks of memory

- Allocating generic blocks of memory—that is, memory not associated with an object—is done by using the standard malloc library of calls
- As is the case with any memory allocated using malloc, there is a responsibility for releasing that memory when are done with it by calling the free function
- The system does not release malloc-based blocks
- Regardless of how memory is allocated memory, managing overall memory usage is important
- iOS has a virtual memory system, it does not use a swap file
 - Code pages can be flushed as needed but an application’s data must all fit into memory at the same time.
 - The system monitors the overall amount of free memory and does what it can to give your application the memory it needs
 - If memory usage becomes too critical though, the system may terminate your application
 - This option is used only as a last resort, to ensure that the system has enough memory to perform critical operations such as receiving phone calls

Be Prepared to Stop – Don't stop programmatically

- iOS applications stop when people press the Home button to open a different application
 - Or use a device feature, such as the phone
- **Save user data as soon as possible and as often as reasonable**
 - An exit or terminate notification can arrive at any time
- **Save the current state when stopping**, at the finest level of detail possible so that people don't lose their context when they start the application again
 - For example, if your app displays scrolling data, save the current scroll position
- Never quit an iOS application programmatically because people tend to interpret this as a crash

Xcode is an integrated development environment (IDE) for creating apps



Dealing with the *NeXT Interface Builder* (Nib) File

- A view controller loads its nib file in its loadView method
- The nib file can also be loaded using an instance of NSBundle
- You can learn more about loading nib files in *Resource Programming Guide*
- If you initialize a view controller using initWithNibName:bundle: but you want to perform additional configuration after the view is loaded
- Override the controller's viewDidLoad method
- The view controller's nib file contains three objects, the File's Owner proxy, the First Responder proxy, and a view
 - You can see them by inspecting the nib file.

To iAd or not iAd

- **iAd** is a mobile advertising platform developed by Apple Inc. for its iPhone
- Allows third-party developers to directly embed advertisements into their applications
- User taps on an iAd banner, a full-screen advertisement appears within the application
- Apple will retain 40% of the ad revenue, the other 60% going to the developers
- In one case with 69K accesses to iAd, the developer received \$16.05

QUESTIONS?
**Please remember your session
evaluation**
Your Feedback is Important to Us



Sources

- **Getting Started Creating an iPhone App**

- http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/Creating_an_iPhone_App/index.html

- **Development Tips**

- <http://www.sitepoint.com/iphone-development-12-tips/>

- **iPhone Apps Design Mistakes**

- <http://www.smashingmagazine.com/2009/11/15/iphone-apps-design-mistakes-disregard-of-context/>

- **Introduction to iPhone App Design**

- <http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html>

- **Introduction to Cocoa Objective-C**

- http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html#//apple_ref/doc/uid/TP30001163

Sources (continued)

- Introduction to Cocoa Objective-C
- http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html#//apple_ref/doc/uid/TP30001163
- User Experience Guidelines
- <http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/UEBestPractices/UEBestPractices.html>