Getting Started with iPhone Development

Tom Harrington
Atomic Bird, LLC
tph@atomicbird.com
Goals
Goals

- Introduction to iPhone Development
Goals

- Introduction to iPhone Development
- Native iPhone apps, not iPhone web apps
Goals

- Introduction to iPhone Development
- Native iPhone apps, not iPhone web apps
- Using official Apple iPhone SDK
Goals

- Introduction to iPhone Development
- Native iPhone apps, not iPhone web apps
- Using official Apple iPhone SDK
- Intended for experienced developers interested in iPhone
iPhone Run-Time Environment
OS Basics
OS Basics

What is the iPhone OS?
OS Basics

- What is the iPhone OS?
- iPhone OS is “OS X”
OS Basics

- What is the iPhone OS?
- iPhone OS is “OS X”
- Essentially, Mac OS X Mobile
OS Basics

- What is the iPhone OS?
- iPhone OS is "OS X"
- Essentially, Mac OS X Mobile
- Not a phone-specific OS
OS Basics

- Cocoa Touch
- Media
- Core Services
- Core OS
Core OS Layer

- BSD Unix kernel
- Drivers
- Virtual Memory
- Threads
- Filesystem
- Networking
Core Services Layer

- Foundation
  - Comparable to C++ STL
- System Configuration
- SQLite
Media Layer

- Audio
- Video
- Graphics

Cocoa Touch
Media
Core Services
Core OS
Cocoa Touch Layer

- User Interface
- High-level Objective-C interfaces to lower layers
iPhone Hardware

- Phone
- Camera
- Microphone

- Used in speaker-phone mode but also available to applications
iPhone + iPod Touch

Hardware

- Accelerometer
  ➢ Detects motion
  ➢ Detects device orientation
- Multi-touch interface
iPhone + iPod Touch

Hardware

- Location info
  - GPS/cell triangulation on iPhone
  - WiFi database for iPod

- Speaker
Application Constraints
Application Constraints

- Limited memory
  - Enforced by system watchdog
  - Filesystem is present, but no swapping
Application Constraints

- Limited memory
  - Enforced by system watchdog
  - Filesystem is present, but no swapping
- Only one application can be active at a time
Application Constraints

- Limited memory
  - Enforced by system watchdog
  - Filesystem is present, but no swapping
- Only one application can be active at a time
- No background processes
Application Constraints

- Limited memory
  - Enforced by system watchdog
  - Filesystem is present, but no swapping
- Only one application can be active at a time
- No background processes
- No copy/paste
Code Signing

- Applications must be digitally signed before the OS will permit them to run
- Each developer needs their own key
Sandboxing
Sandboxing

- iPhone OS enforces “sandboxing” of applications
Sandboxing

- iPhone OS enforces “sandboxing” of applications
- Applications can only write files in their own designated file space
Sandboxing

- iPhone OS enforces “sandboxing” of applications
- Applications can only write files in their own designated file space
- Paths to these spaces include UUIDs, to prevent guessing
Sandboxing

- iPhone OS enforces "sandboxing" of applications
- Applications can only write files in their own designated file space
- Paths to these spaces include UUIDs, to prevent guessing
- Applications even have their own /tmp/
Application Architecture
Model-View-Controller

View
User Interface

Controller
Mediator

Model
Data
Common MVC pattern

View
User Interface

Controller
Mediator

Model
Data
Common MVC pattern

- UI Elements don’t store data, they request it from a controller when needed.
Common MVC pattern

- UI Elements don’t store data, they request it from a controller when needed.
- Controller responds to UI requests and coordinates with model class(es).

![Diagram showing MVC pattern with View, Controller, and Model components: User Interface, Mediator, and Data.]
Model-View-Controller
Model-View-Controller

- Cocoa is designed with MVC in mind
- Not required to use MVC, but following Cocoa’s lead makes development easier
- Many view and controller classes are provided
- Nearly all UI elements are designed to have a controller “driving” them
Cocoa
Cocoa

“Cocoa” is the name for Apple’s library of high-level classes and functions.
Cocoa

- “Cocoa” is the name for Apple’s library of high-level classes and functions
- Written in Objective C
Cocoa

“Cocoa” is the name for Apple’s library of high-level classes and functions

Written in Objective C

Includes “Foundation”, roughly equivalent to C++ STL
Cocoa

“Cocoa” is the name for Apple’s library of high-level classes and functions

- Written in Objective C
- Includes “Foundation”, roughly equivalent to C++ STL
- Includes UI classes
Cocoa MVC Example

Table view

Controller

Model
Cocoa MVC Example

Table view

Controller

Model
Cocoa MVC Example

Table view

Need number of rows

Controller

Model
Cocoa MVC Example

Table view

Controller

Model

Need number of rows

Number of rows
Cocoa MVC Example

Table view

Controller

Model
Cocoa MVC Example

Table view

Controller

Model

Need row 1 data
Cocoa MVC Example

Table view

Need row 1 data

Row 1 data

Controller

Model
Cocoa MVC Example

Table view

Controller

Model

Need row 1 data
Row 1 data
Need row 2 data
Cocoa MVC Example

Controller

Model

Table view

- Need row 1 data
- Row 1 data
- Need row 2 data
- Row 2 data
Cocoa MVC Example

Table view

Controller

Model

Need row 1 data

Row 1 data

Need row 2 data

Row 2 data

..
Cocoa MVC Example

Table view

Need row 1 data

Row 1 data

Need row 2 data

Row 2 data

Reload Data

Controller

Model
Cocoa MVC Example

Button

target

Controller

Performs button’s action
Cocoa MVC Example

- Button configuration includes appearance, target object

  ![Diagram]

  - Button
  - Target
  - Controller
  - Performs button’s action
Cocoa MVC Example

- Button configuration includes appearance, target object
- Responder code goes in a controller class
Cocoa MVC Example

- Button configuration includes appearance, target object
- Responder code goes in a controller class
- Button does not handle clicks, controller does
Application Architecture

UIApplication
Application Architecture

UIApplication

Views

Controllers

Models
Application Architecture

UIApplication

Views

Controllers

Models

Application Bundle

Executable
Images

Data files

Anything else app needs
Development Environment
Develop on a Mac

- Apple provides free iPhone developer tools
- Same tools as for Mac OS X development
- Use any Intel Mac
Objective-C
Objective-C

- Cocoa is written in Objective-C
Objective-C

- Cocoa is written in Objective-C
- Your applications will be too
Objective-C

- Cocoa is written in Objective-C
- Your applications will be too
- At least partially
  - C++ can be used
  - UI and some controller classes will need to be Objective-C
What is Objective-C?
What is Objective-C?

- Object-Oriented extension of C
What is Objective-C?

- Object-Oriented extension of C
- Superset of C, so all C code is valid Objective-C
What is Objective-C?

- Object-Oriented extension of C
- Superset of C, so all C code is valid Objective-C
- Dynamic object typing and method dispatch
What is Objective-C?

- Object-Oriented extension of C
- Superset of C, so all C code is valid Objective-C
- Dynamic object typing and method dispatch
- Single Inheritance
What is Objective-C?

- Object-Oriented extension of C
- Superset of C, so all C code is valid Objective-C
- Dynamic object typing and method dispatch
- Single Inheritance
- OO features based on Smalltalk
Dynamic Typing

"An object is as an object does"
Dynamic Typing

- “An object is as an object does”
- Which methods are implemented is often more important than class hierarchy
Dynamic Typing

- “An object is as an object does”
- Which methods are implemented is often more important than class hierarchy
- Introspection and reflection support this
Dynamic Typing

- “An object is as an object does”
- Which methods are implemented is often more important than class hierarchy
- Introspection and reflection support this
- Objects can be asked if they implement a specific method
If it quacks like a duck...
If it quacks like a duck...
Dynamic Typing
Dynamic Typing

- Will often see objects of type “id”
Dynamic Typing

- Will often see objects of type “id”
- “id” is like the OO equivalent of “void *”, and indicates “object of any class”
Dynamic Typing

- Will often see objects of type “id”
- “id” is like the OO equivalent of “void *”, and indicates “object of any class”
- Calling a method that a class does not implement is not a compiler error
Dynamic Typing

- Will often see objects of type “id”
- “id” is like the OO equivalent of “void *”, and indicates “object of any class”
- Calling a method that a class does not implement is not a compiler error
- It might or might not be a run-time error
ObjC Memory Management
ObjC Memory Management

- Based on retain counts
ObjC Memory Management

- Based on retain counts
- New objects have retain count = 1
ObjC Memory Management

- Based on retain counts
- New objects have retain count = 1
- Increase retain count with "retain" method
  - Adding to collection classes does this
ObjC Memory Management

- Based on retain counts
- New objects have retain count = 1
- Increase retain count with "retain" method
  ➢ Adding to collection classes does this
- Decrease with "release" method
ObjC Memory Management

- Based on retain counts
- New objects have retain count = 1
- Increase retain count with “retain” method
  - Adding to collection classes does this
- Decrease with “release” method
- When retain count reaches 0, object is deallocated
Object Life Cycle
Object Life Cycle

 Allocate Add to array Release Remove from array
Object Life Cycle

Allocate Add to array Release Remove from array

Retain Count

0 1 2
Object Life Cycle

Allocate Add to array Release Remove from array

Retain Count

0 1 2
Object Life Cycle

Allocate Add to array Release Remove from array

Retain Count

0 1 2
Object Life Cycle

Allocate Add to array Release Remove from array

Retain Count

0 1 2
Object Life Cycle

Allocate | Add to array | Release | Remove from array | Deallocation
ObjC Memory Management

- No Garbage Collection on iPhone
Method Syntax

- Square-bracket syntax

```objectivec
[myArray addObject:foo];
```

- Arguments interleave with method name

```objectivec
[myArray insertObject:foo atIndex:5];
```

➤ In this case the method name is “insertObject:atIndex:”
Method Declaration Syntax

- (void)insertObject:(id)anObject atIndex:(int)index;
Method Declaration Syntax

- (void)insertObject:(id)anObject atIndex:(int)index;
Method Declaration Syntax

- (void)insertObject:(id)anObject atIndex:(int)index;

Method name  Argument types
Method Declaration Syntax

- (void)insertObject:(id)anObject atIndex:(int)index;
Method Declaration Syntax

- (void)insertObject:(id)anObject atIndex:(int)index;

- Method name
  - Argument types
    - Argument names
  - Return type
Method Declaration Syntax

- (void)insertObject:(id)anObject atIndex:(int)index;

Method type
Return type
Method name
Argument types
Argument names
Method Call Syntax

[myArray insertObject:foo atIndex:5];
Method Call Syntax

```
[myArray insertObject:foo atIndex:5];
```
Method Call Syntax

```
[myArray insertObject:foo atIndex:5];
```

Method name

Arguments
Method Call Syntax

[myArray insertObject:foo atIndex:5];

Method name
Target object
Arguments
Method Call Syntax

- Calls can be nested

```
[myArray
    insertObject:
        [NSString stringWithFormat:@"%d minutes", x]
    atIndex:[[myOtherArray objectAtIndex:4] intValue]];
```
Properties

- Provides a more-familiar “dot” syntax for getter/setter methods

  ```
  [foo setText:@"bar"];  
  is equivalent to
  foo.text = @"bar";
  ```

- Properties imply creation of equivalent Objective-C methods at compile time
Class definitions

- Declarations and definitions are usually separate

- Declaration in a header file
  ```cpp
  @interface MyClass : NSObject
  ...
  @end
  ```

- Definition in an implementation file
  ```cpp
  @implementation MyClass
  ...
  @end
  ```
Class definitions

- Can have multiple @interface and @implementation blocks for the same class, via “categories”

@interface MyClass : NSObject

@interface MyClass (extensions)
Protocols

- Protocol is similar to a Java or C++ interface
- Often used as an alternative to multiple inheritance
Delegates
Delegates

- Many classes allow one object to be set as another object’s “delegate”
Delegates

- Many classes allow one object to be set as another object’s “delegate”
- Delegates receive method calls from the object
Delegates

- Many classes allow one object to be set as another object’s “delegate”
- Delegates receive method calls from the object
- These methods are usually declared in a protocol
Delegates
Delegates

Delegate methods typically serve one of two purposes
Delegates

- Delegate methods typically serve one of two purposes
  - Policy decisions, a “shouldDoSomething” method returning boolean
Delegates

Delegate methods typically serve one of two purposes

- Policy decisions, a “shouldDoSomething” method returning boolean
- Advisory messages, a “didDoSomething” method with information about what was done
Delegates

- Delegate methods typically serve one of two purposes
  - Policy decisions, a “shouldDoSomething” method returning boolean
  - Advisory messages, a “didDoSomething” method with information about what was done
- Delegation is often used instead of subclassing
Delegates: Example
Delegates: Example

- NSURLConnection, a general-purpose URL-loading class
Delegates: Example

- `NSURLConnection`, a general-purpose URL-loading class
- Delegate methods include
Delegates: Example

- **NSURLConnection**, a general-purpose URL-loading class
- Delegate methods include
  - Asking for authentication
Delegates: Example

- NSURLConnection, a general-purpose URL-loading class
- Delegate methods include
  - Asking for authentication
  - Returning connection status (e.g. HTTP response code)
Delegates: Example

- **NSURLConnection**, a general-purpose URL-loading class
- Delegate methods include
  - Asking for authentication
  - Returning connection status (e.g. HTTP response code)
  - Returning data loaded from the connection
Delegates: Example

- NSURLConnection, a general-purpose URL-loading class
- Delegate methods include
  - Asking for authentication
  - Returning connection status (e.g. HTTP response code)
  - Returning data loaded from the connection
  - Notification that connection has finished, or failed
Delegates: Example

My Controller -> Start loading HTTP request -> NSURLConnection
Delegates: Example

My Controller  →  Start loading HTTP request  →  Delegate Methods  →  Response received (HTTP status)  → NSURLConnection
Delegates: Example

My Controller → Start loading HTTP request →NSURLConnection

Delegate Methods
Response received (HTTP status)

Data available
Delegates: Example

My Controller → Start loading HTTP request → NSURLConnection

Delegate Methods

- Response received (HTTP status)
- Data available
- Data available
- Data available
Delegates: Example

My Controller

Start loading HTTP request

Delegate Methods

Response received (HTTP status)

Data available

Data available

Data available

Connection complete

NSURLConnection
Delegates: Example

My Controller ➔ Start loading HTTP request ➔ NSURLConnection

Delegate Methods

Connection failed
Developer Tools
Xcode
Xcode

- Apple’s Integrated Development Environment
Xcode

- Apple’s Integrated Development Environment
- Project management
Xcode

- Apple’s Integrated Development Environment
- Project management
- Compile applications (gcc)
Xcode

- Apple’s Integrated Development Environment
- Project management
- Compile applications (gcc)
- Debug apps (gdb)
Xcode

- Apple’s Integrated Development Environment
- Project management
- Compile applications (gcc)
- Debug apps (gdb)
- Deploys applications to iPhone during development
Interface Builder

- Graphical user interface design tool
Interface Builder

- Graphical user interface design tool
- Provides palette of UI elements
Interface Builder

- Graphical user interface design tool
- Provides palette of UI elements
- Integrates with Xcode
Interface Builder

- Graphical user interface design tool
- Provides palette of UI elements
- Integrates with Xcode
  - Reads source files to find available classes
Interface Builder

- Graphical user interface design tool
- Provides palette of UI elements
- Integrates with Xcode
  - Reads source files to find available classes
  - Finds application resources in project
Interface Builder

- Graphical user interface design tool
- Provides palette of UI elements
- Integrates with Xcode
  - Reads source files to find available classes
  - Finds application resources in project
- Creates object archives
Interface Builder

- Graphical user interface design tool
- Provides palette of UI elements
- Integrates with Xcode
  - Reads source files to find available classes
  - Finds application resources in project
- Creates object archives
  - No code is generated
**Interface Builder**

- Graphical user interface design tool
- Provides palette of UI elements
- Integrates with Xcode
  - Reads source files to find available classes
  - Finds application resources in project
- Creates object archives
  - No code is generated
  - Objects are created and serialized for later use
iPhone Simulator

- Runs iPhone APIs on Mac OS X
- Not a hardware emulator
- Faster than installing on a device
- No camera, SMS, etc.
Developer Tools Demo
Developer Tools Demo

Hello, Tom!

Hello
iPhone Developer Program
iPhone Developer Program

- Step 1: Sign up for a (free) developer account at developer.apple.com
iPhone Developer Program

- Step 1: Sign up for a (free) developer account at developer.apple.com
- Developer tools are free
iPhone Developer Program

- Step 1: Sign up for a (free) developer account at developer.apple.com
- Developer tools are free
- NDA required for pre-release software
iPhone Developer Program

- Step 1: Sign up for a (free) developer account at developer.apple.com
- Developer tools are free
- NDA required for pre-release software
- Simulator only
iPhone Developer Program
iPhone Developer Program

- Paid accounts
  - $99/year for standard account
  - $299 for enterprise account
  - Enterprise accounts can deploy applications internally without going through iPhone app store
iPhone Developer Program

- Paid accounts
  - $99/year for standard account
  - $299 for enterprise account
    - Enterprise accounts can deploy applications internally without going through iPhone app store

- Can’t deploy apps without one
iPhone Developer Program

- Paid accounts
  - $99/year for standard account
  - $299 for enterprise account
    - Enterprise accounts can deploy applications internally without going through iPhone app store

- Can’t deploy apps without one
- Can test on iPhone or iPod Touch
iPhone Developer Program

- Paid accounts
  - $99/year for standard account
  - $299 for enterprise account
  - Enterprise accounts can deploy applications internally without going through iPhone app store

- Can’t deploy apps without one
- Can test on iPhone or iPod Touch
- Applications can take a long time to process
Beta Testing
Beta Testing

“Ad Hoc” distribution
Beta Testing

- “Ad Hoc” distribution
- Up to 100 devices
Beta Testing

- “Ad Hoc” distribution
- Up to 100 devices
- Custom app build that runs only on those devices
Beta Testing

- “Ad Hoc” distribution
- Up to 100 devices
- Custom app build that runs only on those devices
- Users install application using iTunes
Beta Testing

- “Ad Hoc” distribution
- Up to 100 devices
- Custom app build that runs only on those devices
- Users install application using iTunes
- Also useful for client testing
Distributing Applications
Distributing Applications

- Must use Apple’s iPhone Application Store
Distributing Applications

- Must use Apple’s iPhone Application Store
- Available on iPhone or via iTunes
Distributing Applications

- Must use Apple’s iPhone Application Store
- Available on iPhone or via iTunes
- 70% of revenue to developer, 30% to Apple
iPhone Application Store

- Software downloads directly to device
- Automatic install
iPhone Application Store

- Update notification
- Automatic install
- App data preserved
iPhone Application Store

- Update notification
- Automatic install
- App data preserved
iPhone Application Store

- Apps download to desktop
- Sync to device
iPhone Application Store
iPhone Application Store

- Apple tests all applications
iPhone Application Store

- Apple tests all applications
- If app requires signup/account info, Apple will want a test account
iPhone Application Store

- Apple tests all applications
- If app requires signup/account info, Apple will want a test account
- Typical time to approve/reject: 7-10 days
iPhone Application Store

- Apple tests all applications
- If app requires signup/account info, Apple will want a test account
- Typical time to approve/reject: 7-10 days
- Your mileage may vary (significantly)
iPhone Application Store
iPhone Application Store

Apps can be rejected for
iPhone Application Store

- Apps can be rejected for
  ➢ Crashes
Apps can be rejected for

➤ Crashes

➤ UI bugs
iPhone Application Store

- Apps can be rejected for
  - Crashes
  - UI bugs
  - Use of undocumented/unsupported APIs
iPhone Application Store

- Apps can be rejected for
  - Crashes
  - UI bugs
  - Use of undocumented/unsupported APIs
  - Other reasons documented in developer program agreement
iPhone Application Store

- Apps can be rejected for
  - Crashes
  - UI bugs
  - Use of undocumented/unsupported APIs
  - Other reasons documented in developer program agreement
  - Other reasons not documented in developer program agreement
iPhone Store Inventory

- Electronic Books: 247
- Business: 289
- Education: 1,021
- Entertainment: 1,190
- Finance: 165
- Games: 546
- Health care/Fitness: 226
- Lifestyle: 231
- Music: 191
- Navigation: 80
- News: 121
- Photography: 147
- Productivity: 564
- Reference: 432
- Social Networking: 158
- Sports: 401
- Travel: 34
- Utilities: 1,049
- Weather: 1,190

Number of Applications

As of 10/5/2008
Resources

- Apple Developer Web Site
  - Extensive documentation
  - Sample code
  - Developer tools

http://developer.apple.com/
Resources

- **Cocoa Programming for Mac OS X**, 3rd ed., Aaron Hillegass
- Save 35% at informit.com with code SWSUMMIT08
Resources

- *Programming in Objective-C*, Stephen Kochan
- 2nd edition due soon
Resources

- The iPhone Developer's Cookbook
- iPhone SDK Development
- Beginning iPhone Development
- iPhone Programmer's Road Map
- iPhone SDK Application Development
Getting Started with iPhone Development

Tom Harrington
Atomic Bird, LLC
tph@atomicbird.com