I apologize to all who are in the overflows. Just 3 hours...
If you haven’t downloaded the iPhone SDK yet, start now!
illini apple center
“The goal of DevPhone is to teach you the basics of native iPhone application programming.”
How are we getting to our goal?

The pieces of DevPhone
How are we getting to our goal?

The pieces of DevPhone

Sessions
How are we getting to our goal?

The pieces of DevPhone

- Labs
- Sessions
How are we getting to our goal?
The pieces of DevPhone

Programming Competition

Labs

Sessions
The Sessions

• A session will be a technical lecture over a specific topic relating to iPhone programming.
• Each will end with a 15 minute Q & A.
• Slides are online.
• I’m assuming you’re familiar with:
  ▪ Programming in general.
  ▪ Object-oriented programming specifically.
Have questions?

You can chat on our web-IRC client

• http://acm.uiuc.edu/macwarriors/devphone/chat
Labs

• Your opportunity to get 1-on-1 help from someone who knows how to program for the iPhone.
• Bring your machine, bring your questions.
• Don’t ask if you haven’t tried it yourself.
| Time   | Session                                                                 | Lab                      |
|--------|--------------------------------------------------------------------------------------------------------------------------------------|
| 1:00 PM| **Keynote — Introduction to programming for the iPhone**<br>SC 2405-1109-1105 / Kevin Cathey / Programming competition starts here | Labs closed              |
| 2:00 PM| **Introduction to Objective-C 2.0**<br>SC 2405-1109-1105 / Kevin Cathey                                                            | Open Hours               |
| 3:30 PM| **Introduction to the Apple Developer Tools**<br>SC 2405-1109-1105 / Kevin Cathey                                                      | SC 2407                  |
| 5:00 PM| **Dinner and special talk**<br>SC 2405 / Karrie Karahalios                                                                          | Labs closed              |
| 6:00 PM| **Looking deeper at Objective-C**<br>SC 1404 / Kevin Cathey                                                                            | Open Hours               |
| 7:30 PM| **Starting to code with the Foundation framework**<br>SC 1404 / Kevin Cathey                                                            | SC 2407                  |
## Sunday Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:00 PM</td>
<td><strong>Building graphical user interfaces with UIKit</strong>&lt;br&gt;SC 1404 / Kevin Cathey</td>
<td><strong>Open Hours</strong>&lt;br&gt;SC 2407</td>
</tr>
<tr>
<td>2:30 PM</td>
<td><strong>Using iPhone hardware: accelerometer, camera, and location services</strong>&lt;br&gt;SC 1404 / Kevin Cathey</td>
<td></td>
</tr>
<tr>
<td>4:00 PM</td>
<td><strong>iPhone graphics: CoreGraphics, OpenGL ES, and CoreAnimation</strong>&lt;br&gt;SC 1404 / Kevin Cathey</td>
<td></td>
</tr>
<tr>
<td>7:00 PM</td>
<td><strong>Programming competition submissions due</strong>&lt;br&gt;at acm.uiuc.edu/macwarriors/devphone/submissions</td>
<td><strong>Labs closed</strong></td>
</tr>
<tr>
<td>8:00 PM</td>
<td><strong>Programming competition presentations and awards</strong>&lt;br&gt;SC 1404 / Kevin Cathey</td>
<td></td>
</tr>
</tbody>
</table>
Programming competition

• Show us you’ve learned something, and get rewarded.
• Rules
  ▪ Work individually or in pairs.
  ▪ Must stick to one of the three options provided.
  ▪ Submit on our website by 7 PM tomorrow (Sunday).
Programming competition

• Judging categories
  ▪ Function
  ▪ Creative use of iPhone technologies
  ▪ Technical achievement
  ▪ Usability and UI design
  ▪ Usefulness

• Starts right now! (well, after the next slide)
Three options
Three options

Option 1 — Boggle
Create the game of Boggle. For each game, have a random board. Select strings of characters to generate a word. Verify that a user-selected word is indeed a valid word. When the timer runs out, the game is over.
Three options

Option 1 — Boggle
Create the game of Boggle. For each game, have a random board. Select strings of characters to generate a word. Verify that a user-selected word is indeed a valid word. When the timer runs out, the game is over.

Option 2 — Course Catalog
By adding “.xml” to any UIUC course catalog page (e.g. http://courses.illinois.edu/cis/2009/spring/schedule/CS/index.xml), you can get XML data instead of HTML data. Use the course catalog data to create a Course Catalog application that allows you to browse classes and see when they occur. Also, you could figure out for any given room, what class is currently going on.
Three options

Option 1 — Boggle
Create the game of Boggle. For each game, have a random board. Select strings of characters to generate a word. Verify that a user-selected word is indeed a valid word. When the timer runs out, the game is over.

Option 2 — Course Catalog
By adding “.xml” to any UIUC course catalog page (e.g. http://courses.illinois.edu/cis/2009/spring/schedule/CS/index.xml), you can get XML data instead of HTML data. Use the course catalog data to create a Course Catalog application that allows you to browse classes and see when they occur. Also, you could figure out for any given room, what class is currently going on.

Option 3 — Recipe Book
Create a recipe book application. Add a recipe with ingredients, baking instructions, and a picture. To get the picture, you could use the phone’s camera. Or how about dynamically scaling the ingredients based upon how many you are serving?
Prizes

First Place
$200 gift certificate to the Illini Apple Store.

Runner-up
$50 gift certificate to the Illini Apple Store.

(prizes are per pair if working in a pair)
Introduction to iPhone Programming
Introduction to iPhone Programming

acm.uiuc.edu/macwarriors/devphone
Introduction to iPhone Programming

What’s in an iPhone application?

acm.uiuc.edu/macwarriors/devphone
Introduction to iPhone Programming

What’s in an iPhone application?

What happens when I run an iPhone application?

acm.uiuc.edu/macwarriors/devphone
Introduction to iPhone Programming

- What’s in an iPhone application?
- What happens when I run an iPhone application?
- Considerations for iPhone programming
Introduction to iPhone Programming

- What’s in an iPhone application?
- What happens when I run an iPhone application?
- Considerations for iPhone programming
- What’s the development cycle?

acm.uiuc.edu/macwarriors/devphone
Introduction to iPhone Programming

What’s in an iPhone application?

What happens when I run an iPhone application?

Considerations for iPhone programming

What’s the development cycle?
What’s in an iPhone application?
What’s in an iPhone application?

Called the application bundle → Remote.app
What’s in an iPhone application?

Remote.app
What’s in an iPhone application?

Remote.app

Remote
Binary executable
What’s in an iPhone application?

Remote.app

- Remote
  - Binary executable
- User interface files
What’s in an iPhone application?

Remote.app

- Remote: Binary executable
- User interface files
- Info.plist: Application info and configuration
What’s in an iPhone application?

- Remote.app
  - Remote
    - Binary executable
  - User interface files
  - Info.plist
    - Application info and configuration
  - Images and other resources
The Executable

• Write code in Objective-C.
• Compile for ARM architecture (or x86 if on simulator).
• Link with system libraries.
• These steps done from Xcode.
User interface files
User interface files

- Each application has at least one interface file (called a Nib file).
User interface files

• Each application has at least one interface file (called a Nib file).
• Built with Interface Builder.
User interface files

• Each application has at least one interface file (called a Nib file).
• Built with Interface Builder.
• Contain “freeze-dried” user interface components.
User interface files

• Each application has at least one interface file (called a Nib file).
• Built with Interface Builder.
• Contain “freeze-dried” user interface components.
• At runtime, “unfrozen” and placed in memory with executable code.
The Main Nib File
The Main Nib File

• Remember: just “frozen” objects.
The Main Nib File

• Remember: just “frozen” objects.
• One of these objects is your main window.
The Main Nib File

• Remember: just “frozen” objects.
• One of these objects is your main window.
• For now, think of the Main Nib file as your interface (I’ll show how to load additional Nib files later).
The Main Nib File

• Remember: just “frozen” objects.
• One of these objects is your main window.
• For now, think of the Main Nib file as your interface (I’ll show how to load additional Nib files later).
• Interface elements not in your main Nib file:
  ▪ Status bar
  ▪ Application instance (we’ll talk about proxy objects later)
Info.plist

- Allows you to specify:
  - Information about your application the system uses.
  - Customizations for status bar, protocols, etc.
- In PLIST (property list) format.
Property Lists
Property Lists

- Used frequently for data serialization on OS X.
Property Lists

- Used frequently for data serialization on OS X.
- XML or binary files.
Property Lists

• Used frequently for data serialization on OS X.
• XML or binary files.
• Represents an object graph.
Property Lists

• Used frequently for data serialization on OS X.
• XML or binary files.
• Represents an object graph.
• When loaded by Cocoa [Touch], data in property list turned into real objects.
Property Lists

• Used frequently for data serialization on OS X.
• XML or binary files.
• Represents an object graph.
• When loaded by Cocoa [Touch], data in property list turned into real objects.
• Use property list editor in Xcode to create and edit.
Info.plist
Info.plist

• Required information:
  - Display name (`CFBundleDisplayName`)
  - Identifier (`CFBundleIdentifier`)
  - iPhone application flag (`LSRequiresIPhoneOS`)
  - Main Nib file name (`NSMainNibFile`)
Info.plist

• Required information:
  ▪ Display name (**CFBundleDisplayName**)
  ▪ Identifier (**CFBundleIdentifier**)
  ▪ iPhone application flag (**LSRequiresIPhoneOS**)
  ▪ Main Nib file name (**NSMainNibFile**)

• Customization properties
  ▪ Status bar style (**UIStatusBarStyle** or **UIStatusBarHidden**)
  ▪ Homescreen icon (**CFBundleIconFile**)
  ▪ Others…
Other Resources

• Images (app icon, etc).
• Settings bundle to be showing in Settings application.
• Localizations.
Other Resources

- Images (app icon, etc).
- Settings bundle to be showing in Settings application.
- Localizations.

<table>
<thead>
<tr>
<th>Settings.bundle</th>
<th>Preference pane for the Settings app</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icon-Settings.png</td>
<td>Icon for settings application</td>
</tr>
<tr>
<td>Default.png</td>
<td>The image to show while your app is launching</td>
</tr>
<tr>
<td>en.lproj, fr.lproj, ...</td>
<td>Localized folders (ignore these for now)</td>
</tr>
</tbody>
</table>

Other items… …that you should ignore for now.
The pieces of a running application

The important objects in memory
The pieces of a running application

The important objects in memory

- **Application instance**
  - Controls your application’s interaction with the system.
  - Provided by system by default.
The pieces of a running application
The important objects in memory

• **Application instance**
  - Controls your application’s interaction with the system.
  - Provided by system by default.

• **Application delegate**
  - A class you write.
  - Responds to application instance (memory warnings, etc).
The pieces of a running application
The important objects in memory

• Application instance
  ▪ Controls your application’s interaction with the system.
  ▪ Provided by system by default.

• Application delegate
  ▪ A class you write.
  ▪ Responds to application instance (memory warnings, etc).

• User interface
  ▪ Loaded from Nib files.
The pieces of a running application
The important objects in memory

• **Application instance**
  ▪ Controls your application’s interaction with the system.
  ▪ Provided by system by default.

• **Application delegate**
  ▪ A class you write.
  ▪ Responds to application instance (memory warnings, etc).

• **User interface**
  ▪ Loaded from Nib files.

• **Your code**
  ▪ Custom classes: models and controllers.
Introduction to iPhone Programming

What’s in an iPhone application?

What happens when I run an iPhone application?

Considerations for iPhone programming

What’s the development cycle?
Introduction to iPhone Programming

- What’s in an iPhone application?
- What happens when I run an iPhone application?
- Considerations for iPhone programming
- What’s the development cycle?
Application Life-Cycle
How your application lives and dies
Application Life-Cycle
How your application lives and dies

User taps icon on home screen
Application Life-Cycle
How your application lives and dies

User taps icon on home screen

main()
Application Life-Cycle
How your application lives and dies

- User taps icon on home screen
- main()
- UIApplicationMain()
Application Life-Cycle
How your application lives and dies

User taps icon on home screen

main()

UIApplicationMain()
Application Life-Cycle
How your application lives and dies

- User taps icon on home screen
- main()
- UIApplicationMain()
- System requests termination

Event Loop
Application Life-Cycle
How your application lives and dies

1. User taps icon on home screen
2. main()
3. UIApplicationMain()
4. Event Loop
5. System requests termination
6. Application actually terminates
Application Life-Cycle
How your application lives and dies

User taps icon on home screen

main()

UIApplicationMain()

Event Loop

System requests termination

Application actually terminates
main and UIApplicationMain

• main()
  ▪ Just like any other main function (C, C++, etc).
  ▪ Starts application with UIApplicationMain.
  ▪ Don’t change it.
main and UIApplicationMain

• main()
  ▪ Just like any other main function (C, C++, etc).
  ▪ Starts application with UIApplicationMain.
  ▪ Don’t change it.

main.m

#import <UIKit/UIKit.h>

int main(int argc, char *argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];
    return retVal;
}
main and UIApplicationMain

• **main()**
  - Just like any other `main` function (C, C++, etc).
  - Starts application with `UIApplicationMain`.
  - Don’t change it.

```c
#import <UIKit/UIKit.h>

int main(int argc, char *argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];
    return retVal;
}
```
main and UIApplicationMain

• UIApplicationMain()
  ▪ Creates instance of UIApplication class (responsible for actually launching your application and loading user interface).
  ▪ This instance of UIApplication is the application instance.
Event-Handling Cycle
Event-Handling Cycle

• System receives event (like mouse movement)
Event-Handling Cycle

• System receives event (like mouse movement)
• System sends event on to your application instance
Event-Handling Cycle

• System receives event (like mouse movement)
• System sends event on to your application instance
• Application instance forwards event to First Responder, who sends event up through higher-level objects. This is called the **responder chain**, and is an advanced topic.
Event-Handling Cycle

• System receives event (like mouse movement)
• System sends event on to your application instance
• Application instance forwards event to First Responder, who sends event up through higher-level objects. This is called the **responder chain**, and is an advanced topic.
• Take-away
Event-Handling Cycle

- System receives event (like mouse movement)
- System sends event on to your application instance
- Application instance forwards event to First Responder, who sends event up through higher-level objects. This is called the responder chain, and is an advanced topic.
- Take-away
  - Event handling is complicated.
Event-Handling Cycle

• System receives event (like mouse movement)
• System sends event on to your application instance
• Application instance forwards event to First Responder, who sends event up through higher-level objects. This is called the **responder chain**, and is an advanced topic.

• Take-away
  • Event handling is complicated.
  • You can intercept these events to use them.
Event-Handling Cycle

- System receives event (like mouse movement)
- System sends event on to your application instance
- Application instance forwards event to First Responder, who sends event up through higher-level objects. This is called the responder chain, and is an advanced topic.
- Take-away
  - Event handling is complicated.
  - You can intercept these events to use them.
  - Most controls already do this for you.
Introduction to iPhone Programming

What’s in an iPhone application? ✓

What happens when I run an iPhone application? ✓

Considerations for iPhone programming

What’s the development cycle?

acm.uiuc.edu/macwarriors/devphone
Introduction to iPhone Programming

- What's in an iPhone application?
- What happens when I run an iPhone application?
- Considerations for iPhone programming
- What's the development cycle?
Sandboxing
Sandboxing

• iPhone OS does not give your application free reign like it does on OS X.
Sandboxing

• iPhone OS does not give your application free reign like it does on OS X.
• Your application has access to limited part of the file system.
Sandboxing

• iPhone OS does not give your application free reign like it does on OS X.
• Your application has access to limited part of the file system.
• Each application in its own folder or sandbox: something like: /ApplicationRoot/ApplicationID
Sandboxing

• iPhone OS does not give your application free reign like it does on OS X.
• Your application has access to limited part of the file system.
• Each application in its own folder or sandbox: something like: `/ApplicationRoot/ApplicationID`
• This is a security device preventing a single application from destroying your phone or iPod Touch.
Virtual Memory
Virtual Memory

• Virtual memory exists, but not really.
Virtual Memory

• Virtual memory exists, but not really.
• Gives your application full virtual 32-bit address space.
Virtual Memory

• Virtual memory exists, but not really.
• Gives your application full virtual 32-bit address space.
• But does not write volatile pages to disk (flash memory only has so many write cycles).
Virtual Memory

• Virtual memory exists, but not really.
• Gives your application full virtual 32-bit address space.
• But does not write volatile pages to disk (flash memory only has so many write cycles).
• Instead, OS tries to free up nonvolatile memory (such as code pages).
Virtual Memory

- Therefore:
  - Be efficient with memory.
  - You should (must) respond to `applicationDidReceiveMemoryWarning:` method in your application delegate.
Introduction to iPhone Programming

- What’s in an iPhone application? ✓
- What happens when I run an iPhone application? ✓
- Considerations for iPhone programming ✓
- What’s the development cycle?
# Introduction to iPhone Programming

<table>
<thead>
<tr>
<th>Topic</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s in an iPhone application?</td>
<td>✓</td>
</tr>
<tr>
<td>What happens when I run an iPhone application?</td>
<td>✓</td>
</tr>
<tr>
<td>Considerations for iPhone programming</td>
<td>✓</td>
</tr>
</tbody>
</table>

**What’s the development cycle?**
Development Process

Write code and build user interfaces
Development Process

- Write code and build user interfaces
- Build for simulator
Development Process

Write code and build user interfaces

Build for simulator

Install

Simulator
Development Process

Write code and build user interfaces

Build for device

Build for simulator

Install

Device

Simulator
Development Process

- Write code and build user interfaces
- Build for simulator
- Install
- Build for device
- Sign
- CERT

Simulator

Device
Development Process

Write code and build user interfaces

Build for simulator
Build for device

Install
Sign

Device
Simulator
Introduction to iPhone Programming

- What’s in an iPhone application? ✓
- What happens when I run an iPhone application? ✓
- Considerations for iPhone programming ✓
- What’s the development cycle? ✓

acm.uiuc.edu/macwarriors/devphone
Introduction to iPhone Programming

- What’s in an iPhone application? ✓
- What happens when I run an iPhone application? ✓
- Considerations for iPhone programming ✓
- What’s the development cycle? ✓

acm.uiuc.edu/macwarriors/devphone
About These Slides

If you need to reference the content of this presentation, these slides will be posted on our website, listed below:

acm.uiuc.edu/macwarriors/devphone