Introductory Application Development
This Session

• What Is an iPhone Application Really?
• How does it work?
• How does MVC Apply to iPhone Development?
• Anatomy of an iPhone Application
This Session...

- Basic User Interface Element
  - Buttons, Labels, Images
- Interacting with an iPhone Application
  - Buttons, Touches, etc
- How to Control your iPhone Application behaviour
An iPhone Application

- Has a .app extension:
- LyricMate.app
An iPhone application like an OS X application contains multiple things:

- Icon, default & info.plist
- Resources (Sounds, Images, Video, Pictures, databases)
- Code Files (.h & .m)
- .xib files (Interface Builder document)
iPhone Icon

- Add your 57x57 PNG icon to your project
- Rounded Corners + Shine are automatically added

Magic!
• Simply add “default.png” into your application
• Version number
• Icon file name
• Shiny effect?
Code Files

- Compiled into the iPhone executable
- No people can’t steal your code
Remember MVC?

Model

NSObject

View

UIView

Controllers

UIViewController
How Does It All Work?

main()

UIApplicationMain()

UIApplication

Loads Nibs & informs delegate when done

-(void)applicationDidFinishLaunching:(UIApplication*)app

-(void)applicationWillTerminate:(UIApplication*)app

Done For you
Your Goal

- To Tap into Apple’s event loops
- Not to define your own
- Hence known as
  Event Driven Programming
UIApplicationDelegate

- Created by default on all new iPhone Projects
- Automatically creates your first Window for you
- You take over at this point
Huh?

- An iPhone Application is made up of a Window which has a hierarchy of views
- Each of these views can have sub views
- Let's explain slowly
iPhone Application Anatomy
• In the beginning there was nothing...
• Then UIApplication gave us a window
• We took this window & gave it a blue view

```swift
[Window addSubview:blueView];
```
• We created a smaller red view and added it as a subview to the blueview

```objective-c
[blueview addSubview:redView];
```
How Did We Do that?

• By Using a combination of UIView & UIViewController
• By Putting one “thing” on top of another
• Let’s Do it together in code!!
Navigation Controllers

- Adding view over view over view over view over view over view over view... you get the point
- Navigation Controllers are the answer!!
Navigation Controllers

- Instead of adding a subview, we push and pop a view from the display.
- Think of the nav controller as an array holder for your views traversing when needed.
Problem...

• What if we want somebody to have access to a particular view (say settings) at any given time
Solution

- The Tab Bar (**UITabBarController**)  
- Each “Tab” holds a view
Applications Mix & Match

You were a child
Crawling on your knees toward it
Making Mama so proud
But your voice was too loud
We tried to watch you laughing
You picked the insects off plants
No time to think of consequences

Control yourself
Take only what you need from it
A familiy of foes wanting to be haunted

Control yourself
Take advantage now, and soon.

You change your mind
Like a girl changes clothes
Yeah you, PMS
Like a bitch, I would know
And you over think
Always speak cryptically
I should know
That you’re no good for me

Cause you’re hot then you’re cold
You’re yes then you’re no
You’re in then you’re out
You’re up then you’re down
You’re wrong when it’s right
It’s black and it’s white
We fight, we break up

LyricMate 1.1
Developed By
Brock Woot
Nicholas Ciricosta

Clear All Saved Lyrics
We Will Revisit Those Later
Basic Interface Elements
A Label

- Remember Hello World?
- We can change a label's text via code

@property(nonatomic, copy) NSString *text
How do we connect to it?

- By declaring an instance of it as an IBOutlet
- IBOutlet allows us to “Connect” a user interface object from IB to our Code
- Simply add “IBOutlet” in front of the property

```swift
@property (nonatomic, retain) IBOutlet UILabel *helloLabel;
```
Connecting

Control + Drag
Setting the Text

• Now we can call the label and set the text

```swift
helloLabel.text = "Hello World FROM CODE!!!"
```
Setting the Text in Code

- Lets edit our Hello World app so that now we can set the text remotely from the code
Ok So we Display Text?
Can we Receive It?
UITextField

- Allows us to enter text via the On Screen keyboard
- Optional delegate methods to get the text
- Properties as well if we want to use them
- On Screen keyboard is automatically activated as needed
Nicholas Cirrosta is a noob

www.nickcirrosta.com W00T
When Its Pushed

- MUST REASSIGN FIRST RESPONDER

- (BOOL)textFieldShouldReturn:
  (UITextField *)theTextField
  {
    [theTextField resignFirstResponder];
    [something DoSomething:[theTextField text];
    return YES;
  }
Let's Interact
Stylus?

- HELL NO!!!!
- iPhone Interaction is done with your Human Stylus
Buttons

- Buttons are Miniature UIViews that respond to being touched and can be hooked up to a function call.
Code Or IB?

- Like everything in iPhone OS there are many ways to do the same task.
- Buttons can be created in either Interface Builder or direct in code.
- For now let's just handle it in IB.
Buttons & IBActions

- IBActions are a special kind of function that can be connected to an Interface Builder created Button

-(IBAction)buttonWasPressed;

Simple Really
What Does That Do?

- Once declared in the code as an IBAction, IB will now allow us to drag and connect a button to that function.

- Other direction then connecting the label.
Let's Try That Out!!

• An App that when the button is pressed, the text changes in a label?
The Touch

- Touches are the single most important thing in most iPhone Applications
- Relatively simple to understand
- Can get VERY VERY tricky with MultiTouch
The iPhone Panel

- 320x 480 Display
- 153,600 Different Touch points
The 4 Golden Methods

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
What to do with them?

- Implement them in your UIView Subclass
- Implement them in your UIViewController
- Responder Chain
Who Gets Them?
Whats Inside?

- NSSet
- UIEvent
UIEvent

- Holds the instances of event
- Think of it as a Container holding all the information you need

- (NSSet *)allTouches;
- (NSSet *)touchesForWindow:(UIWindow *)window;
- (NSSet *)touchesForView:(UIView *)view;
What's in the NSSet?

- The NSSet holds the "Touches" themselves in instances of UITouch
What’s a Touch?

@property NSTimeInterval timestamp;
@property UITouchPhase phase;
@property NSUInteger tapCount;
@property UIWindow *window;
@property UIView *view;

- (CGPoint)locationInView:(UIView *)view;
- (CGPoint)previousLocationInView:(UIView *)view;
Your Turn

- Let's create a simple Application that when a view is touched, the background changes colour.

END OF DAY 1
Can I?

- Let's create a simple application called "TouchEvents"
- Tell us when an event is happening
Ok So what?

- Ok so you can make text highlight...
- Remember this App we wrote?
Lets expand it

- Allow the square to move with our finger
- Will need to use a CGPoint
- Lets do it together

```
UITouch *touch = [[event allTouches] anyObject];
```
Challenge

- I have no given you all the information you need to make it work.
- Think about the logic
- Check the Documentation !!!!
Questions?