

EECS 395: Development for Mobile Devices

Instructor: Nathan Nichols

Faculty Advisor: Kristian Hammond

First day

- Introductions/course overview/handout
- Context & History
- Development environment
- Model-View-Controller
- Objective-C

Introductions/
Course overview/
Handout

History & Context

- You will learn how to develop iPhone applications
- This requires learning two (probably new) technologies:
 - Objective-C (syntax/language)
 - Cocoa Touch/UIKit (API/framework/functions)
- This is like learning Japanese
 - Conjugating verbs, etc. (syntax/language)
 - What the verbs actually are (functions)

Bootstrapping problem

- There is a lot of stuff to learn, and we only really have nine weeks
- How do you be productive in an environment while learning a language, its syntax, its APIs, its development environments, etc.?
- By just diving in and doing it

Assignment I

- Handout
- Due next Monday
- HelloWorld

History

- Objective-C was developed by Brad Cox and Tom Love in the early 80s
- Strict superset of C (every C program is also an Objective-C program)
- Originally just a C preprocessor

History

- In the early 80's people were getting excited about OOP, but Smalltalk was slow and unwieldy
- In 1988, Steve Jobs forms NeXT after getting booted from Apple
- NeXT licensed Objective-C and started building it out
- In 1996 Apple bought NeXTStep, OSX, Steve Jobs, and Objective-C
- “C is a language for writing Unix. Objective-C is a language for writing OSX apps.”

History

- Since iPhones are running OSX, it is also a language for writing iPhone apps.
- Why can't Apple use a normal language (C++/C#/Java) like everyone else?
 - More “powerful” than similar languages
 - At this point, they can expect people to come to them
 - Inertia

History

- Why should you have to learn it?
- Learning different language and language paradigms is Good for CS students
- It lets you write iPhone/OSX apps

Development Environment

- There are two ways to program: with an Integrated Development Environment (IDE) or without
- You will use an IDE called XCode, and its best friend, Interface Builder
- Later on in the course, you will use XCode's overachieving cousin, Instruments
- IDEs are Good
- You will learn to love XCode

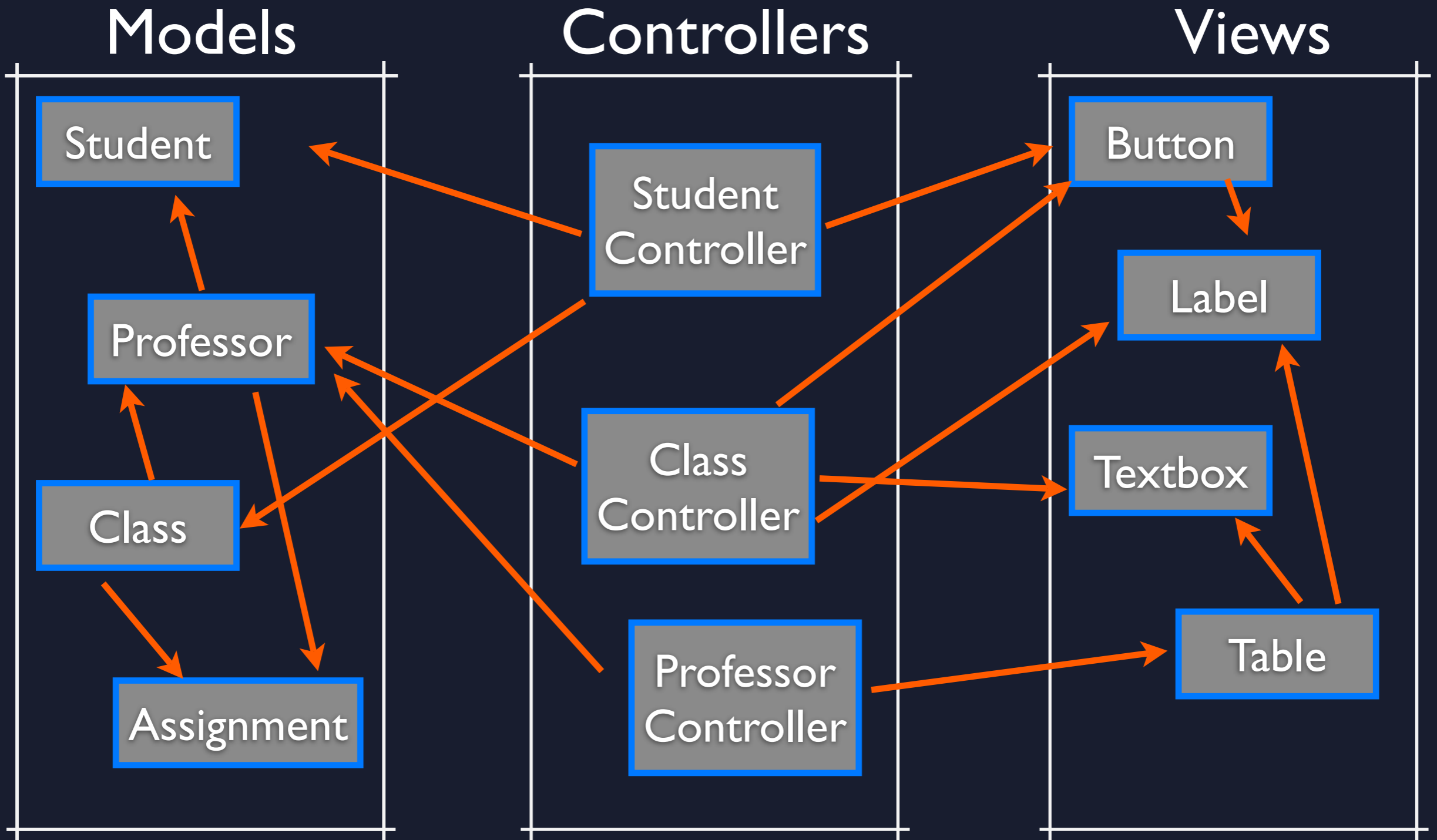
Model-View-Controller

- Because Apple provides the language, the frameworks, and the IDE, they also get to dictate how you setup your applications and your code
- They have decided you will use the Model-View-Controller paradigm
- Fortunately, this is a really good decision

Model-View-Controller

- MVC is generally regarded as a Good Idea
- MVC is NOT Objective-C/Cocoa/iPhone/Apple specific
- Common all over web (Ruby on Rails, Django, Drupal, Silverlight, etc.)
- We will come back to this again and again, but really easy idea

MVC



Big Idea

Models and Views scale ACROSS applications.
Only the Controllers are application specific.

For example, you may never have to write your own View.
Apple has written a bunch of them already. The Button class you use in your application is the same Button class Apple uses in its iPhone apps, and the same Button class I use in mine.

Model-View-Controller

- This “Separation of Concerns” is good for everyone
- You and I don’t have to do our own Button
- Users don’t have to figure out our Button
- Your coworker programming the models doesn’t have to worry about the Button
- In Cocoa Touch, Views inherit from UIView and Controllers inherit from UIViewController

Objective-C

- Doesn't have real namespaces

```
import java.util.Math;  
System.out.println(java.util.Math.abs(-42));
```

- Uses two-letter prefixes instead

```
#import <UIKit/UIKit.h>;  
NSString *name = @"Nate";  
UITableViewCell *cell = [tableView \  
dequeueReusableCellWithIdentifier:CellIdentifier];
```

Objective-C

- Object-oriented, but not as serious about it as Java or C#
- Split interface and implementation between .h and .m files
- Classes always always always inherit from NSObject

Animal.h

```
#import <Foundation/Foundation.h>
```

```
@interface Animal : NSObject {  
    NSString *name;  
    NSString *sound;  
}
```

```
-(id)initWithName:(NSString *)aName sound:(NSString *)aSound;
```

```
-(NSString *)name;  
-(void)setName:(NSString *)aName;  
-(NSString *)sound;  
-(void)setSound:(NSString *)aSound;
```

```
@end
```

Methods

```
-(id)initWithName:(NSString *)aName sound:(NSString *)aSound
```

Method
returns
generic type
“id”.

Method's name is
initWithName:sound:
(seriously)

Method takes two string
arguments

```
Animal *cow = [[Animal alloc] initWithName:@"Cow" sound:@"Moooooo"];
```