

# Developing a photo app for iPhone

Download the app source from:

<https://github.com/raycmorgan/SamplePhotoApp>

# Objective-C

- Superset of C
- Some new syntax
- Classes (OOP)

# New Syntax

@interface, @property, @...

Method syntax

id

Blocks: ^ void(int num) {}

# Classes

```
// MyClass.h  
@interface MyClass : NSObject  
@end
```

```
// MyClass.m  
#import "MyClass.h"  
  
@implementation MyClass  
@end
```

# Classes – Methods

```
// MyClass.h
@interface MyClass : NSObject
- (void)sayHello;
@end
```

```
// MyClass.m
#import "MyClass.h"
@implementation MyClass

- (void)sayHello
{
    // Code goes here
}

@end
```

# Classes – Properties

```
// MyClass.h
@interface MyClass : NSObject
@property (nonatomic, retain) NSString *name;
@end
```

```
// MyClass.m
#import "MyClass.h"
@implementation MyClass
@synthesize name = _name;
@end
```

# Classes – Create Instance

```
MyClass *instance = [[MyClass alloc] init];
```

# Classes – Set Property

```
MyClass *instance = [[MyClass alloc] init];  
instance.name = @"Alex Coleman";
```



# Classes – Call Method

```
MyClass *instance = [[MyClass alloc] init];  
instance.name = @"Alex Coleman";  
[instance sayHello]; // like instance.sayHello();
```

# Classes – Complex Methods

Defining the method:

```
– (void)updatePrice:(RUPrice *)price  
    description:(NSString *)description  
    completionHandler:(RUErrorCallback)completionHandler;
```

Calling the method:

```
[self.item updatePrice:self.price  
    description:self.descriptionField.text  
    completionHandler:^(NSError *error) {  
        // Do stuff after it is saved  
    }];
```

# Classes – Complex Methods

```
[self.item updatePrice:self.price  
                description:self.descriptionField.text  
                completionHandler:^(NSError *error) {  
                    // Do stuff after it is saved  
                }];
```

In another language (JS) it might look like:

```
this.item.update(price, description, function (err) {  
    // Do stuff after it is saved  
});
```

# Cocoa Touch

Apple's framework for helping developers make awesome iOS apps.

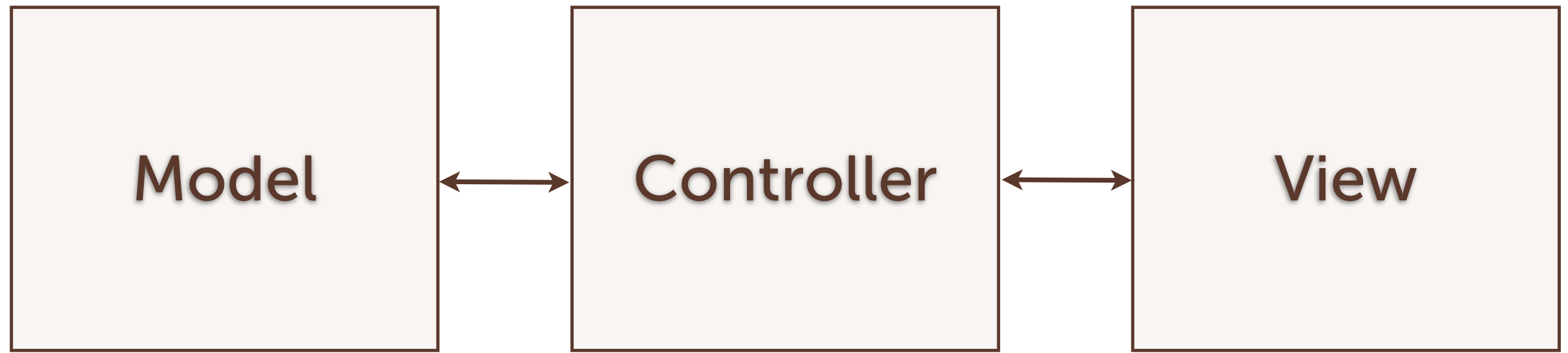
Warning:

Lots of text coming up. Cocoa is big, but just try to remember the key points.

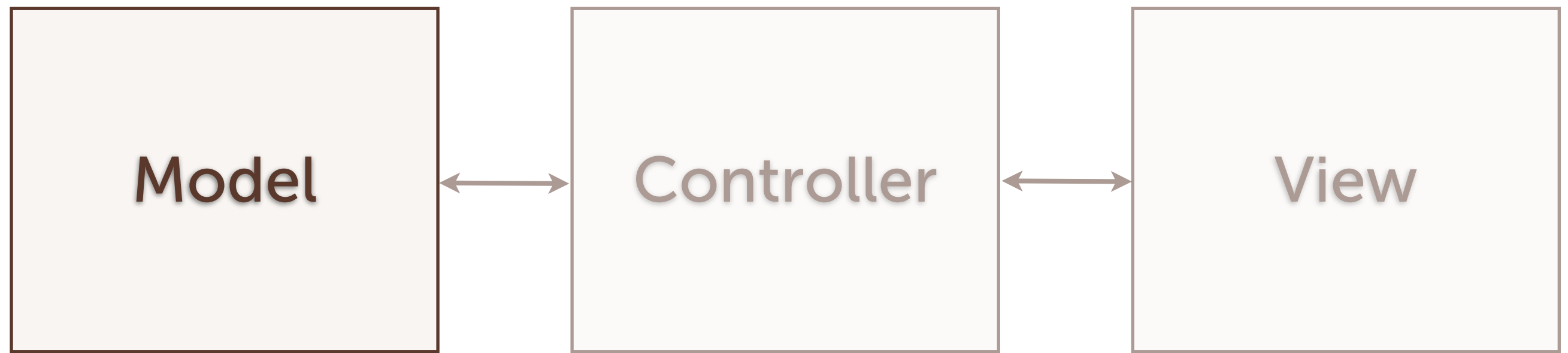
# Cocoa Touch – MVC

- Model
- View
- Controller

# Cocoa Touch – MVC



# Cocoa Touch – Model



- Responsible for all the application data.
- Apple provides Core Data for managing models with disk persistence.

# Cocoa Touch – View



- Responsible for displaying stuff to the screen.
- Should not maintain “state”.



# Cocoa Touch – View



- Responsible for app flow and getting state from the models to the views for displaying.
- Manages user events: taps, pitches, etc.
- On iPhone, one entire screen is managed by one controller (ViewController).

# Cocoa Touch – Delegate Pattern

From Wikipedia:

...an object, instead of performing one of its stated tasks, delegates that task to an associated helper object.

# Cocoa Touch – Delegate Pattern

Me:

A pattern of splitting a single large, highly specialized class, into a smaller general class that use other tiny specialized classes to handle application specific tasks.

# Cocoa Touch – Delegate Pattern

Basically:

Apple provides general classes to do the heavy lifting.

You provide small simple classes to implement your apps specific logic.

# Cocoa Touch – UIViewController

Apple provided base class for managing view hierarchies for a single independent screen.

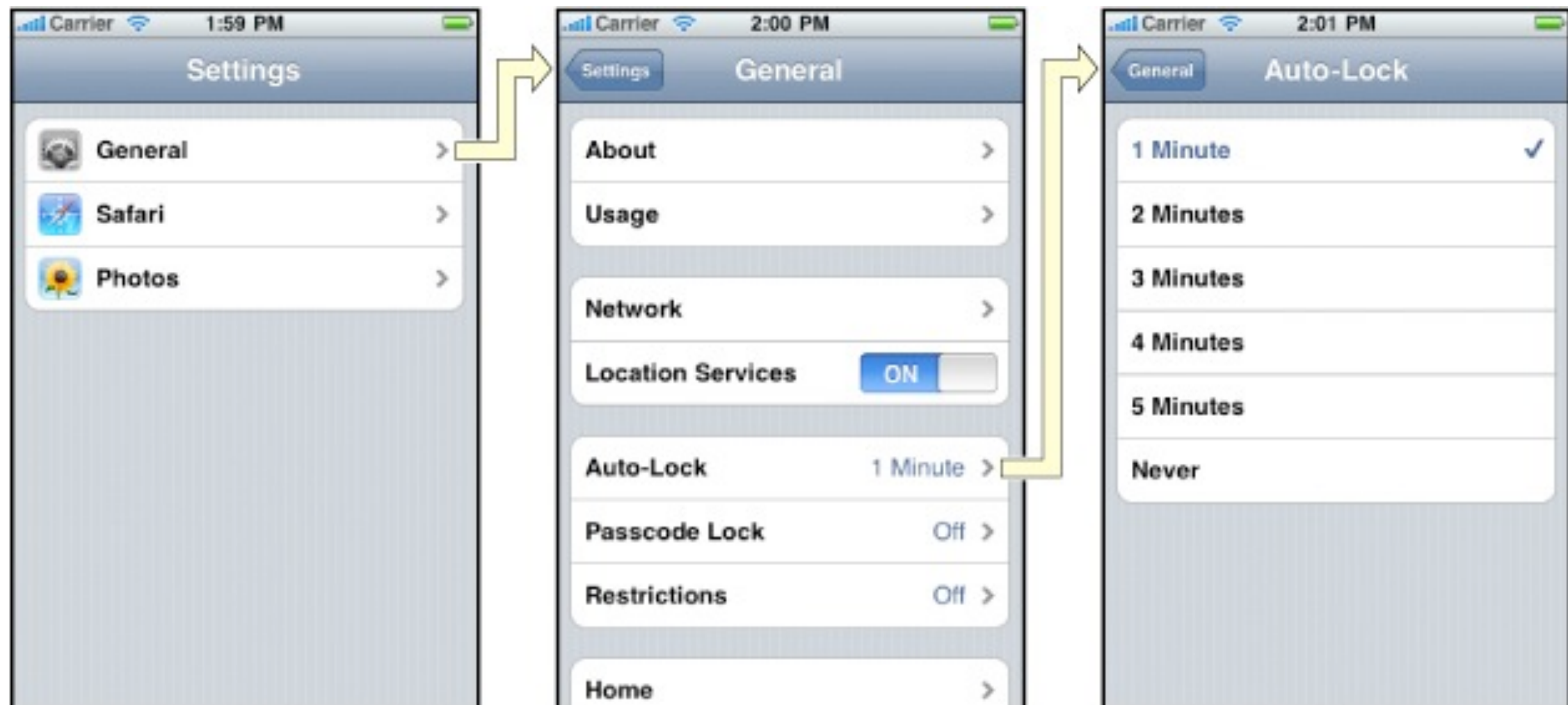
These handle not only keeping what the user sees up to date, but also handle user input.

# Cocoa Touch – Containment Classes

- UINavigationController
- UITabBarController

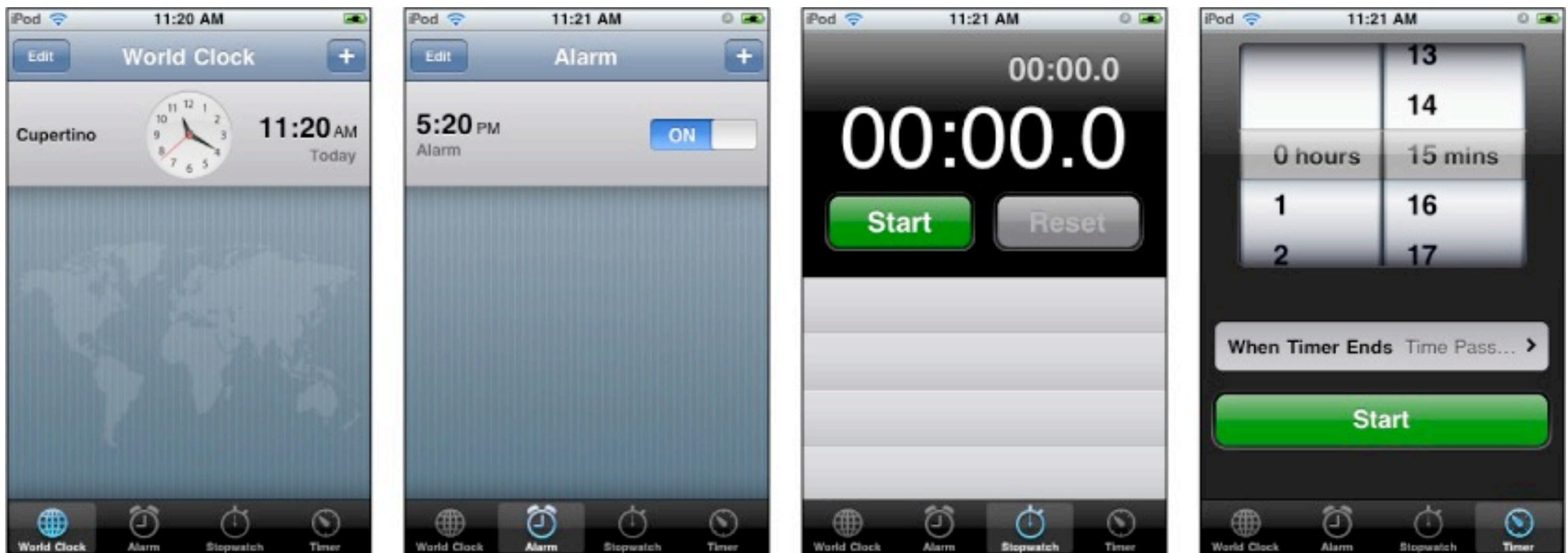
# Cocoa Touch – UINavigationController

- A stack of ViewControllers.
- Provides that “sliding” effect, back button, titles, etc all for you.



# Cocoa Touch – UITabBarController

- A list of ViewControllers.
- Provides the bottom tab bar for switching between the different controllers.





# Cocoa Touch

A rule:

If you are doing something with Cocoa that seems complicated, you are probably doing it wrong.

# Q&A

# Lets build an App!

# References

- iTunes U Stanford iOS class
- WWDC videos from 2010 and 2011
- In #vegastech

@bffmike

@raycmorgan

@phun

# Thanks

Bonus:

Download the source from:

<https://github.com/raycmorgan/SamplePhotoApp>

Want to hack some stuff? Let's hang out and add more to this.

- Save image to disk to see it on next open
- Add a text field for a caption
- Save multiple photos?
- Core Data?
- Post online...