

CS193P - Lecture 4

iPhone Application Development

Building an Application

Model, View, Controller

Nib Files

Controls and Target-Action

Announcements

- Assignment 2
 - Due tomorrow
- Class list:
 - cs193p-auditors@lists.stanford.edu
- WWDC09 Student Scholarship
 - Apply today
 - Due tomorrow!
 - <http://developer.apple.com/wwdc/students/>

Today's Topics

- Application Lifecycle
- Model, View, Controller design
- Interface Builder and Nib Files
- Controls and Target-Action
- HelloPoly demo

Review

Memory Management

- Alloc/Init
 - -alloc assigns memory; -init sets up the object
 - Override -init, not -alloc
- Retain/Release
 - Increment and decrement retainCount
 - When retainCount is 0, object is deallocated
 - Don't call -dealloc!
- Autorelease
 - *MAGIC*
 - Object is released next time through RunLoop

Setters, Getters, and Properties

- Setters and Getters have a standard format:
 - `(int)age;`
 - `(void)setAge:(int)age;`
- Properties allow access to setters and getters through dot syntax:

```
@property age;
```

```
int theAge = person.age;
```

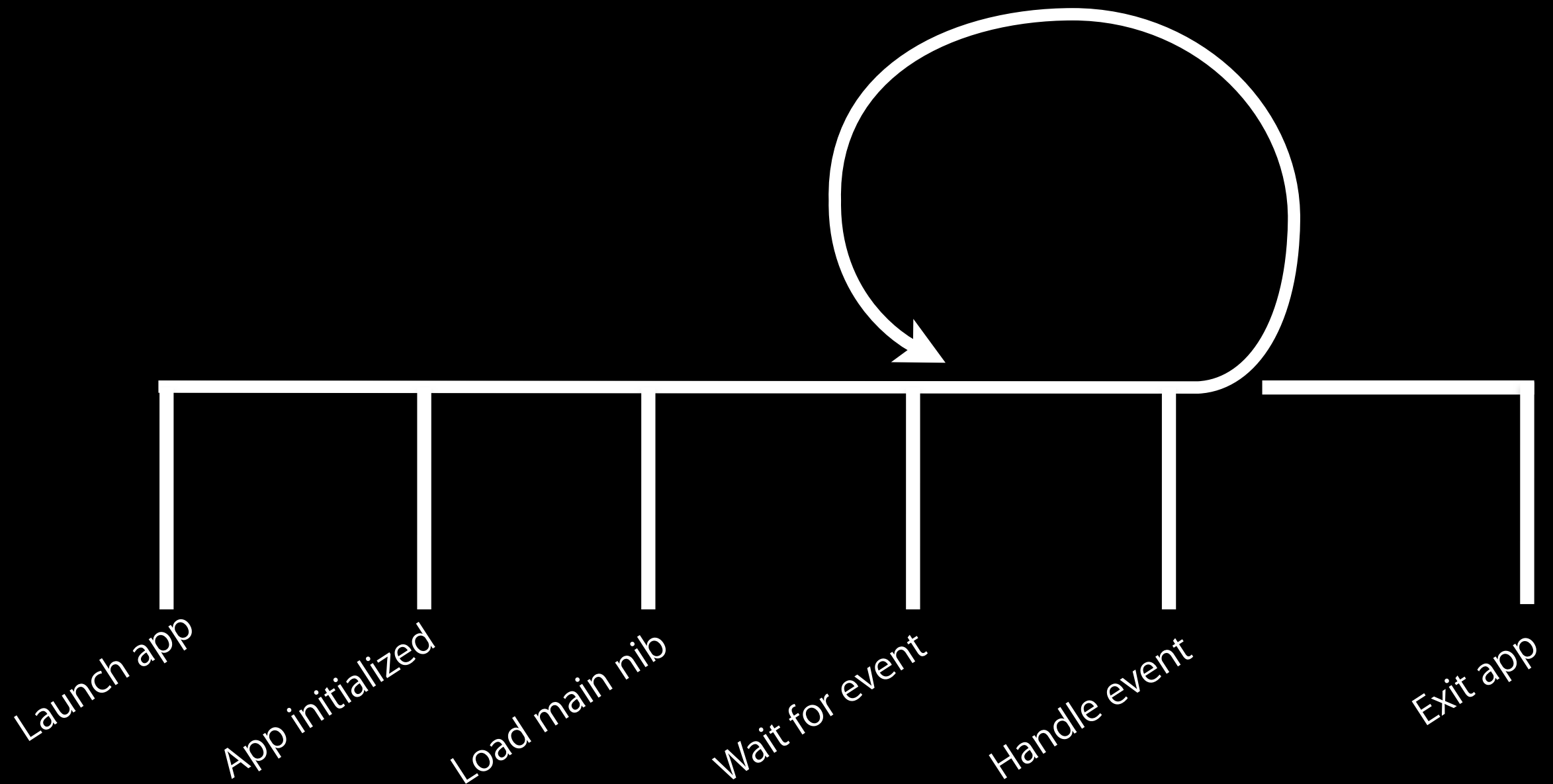
```
person.age = 21;
```

Building an Application

Anatomy of an Application

- Compiled code
 - Your code
 - Frameworks
- Nib files
 - UI elements and other objects
 - Details about object relationships
- Resources (images, sounds, strings, etc)
- Info.plist file (application configuration)

App Lifecycle



UIKit Framework

- Provides standard interface elements
- UIKit and you
 - Don't fight the frameworks
 - Understand the designs and how you fit into them

UIKit Framework

- Starts your application
- Every application has a single instance of UIApplication
 - Singleton design pattern

```
@interface UIApplication  
+ (UIApplication *)sharedApplication  
@end
```

- Orchestrates the lifecycle of an application
- Dispatches events
- Manages status bar, application icon badge
- Rarely subclassed
 - Uses delegation instead

Delegation

- Control passed to delegate objects to perform application-specific behavior
- Avoids need to subclass complex objects
- Many UIKit classes use delegates
 - UIApplication
 - UITableView
 - UITextField

UIApplicationDelegate

- Xcode project templates have one set up by default
- Object you provide that participates in application lifecycle
- Can implement various methods which UIApplication will call
- Examples:
 - (void)applicationDidFinishLaunching:(UIApplication *)application;
 - (void)applicationWillTerminate:(UIApplication *)application;
 - (void)applicationWillResignActive:(UIApplication *)application;
 - (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url;
 - (void)applicationDidReceiveMemoryWarning:(UIApplication *)application;

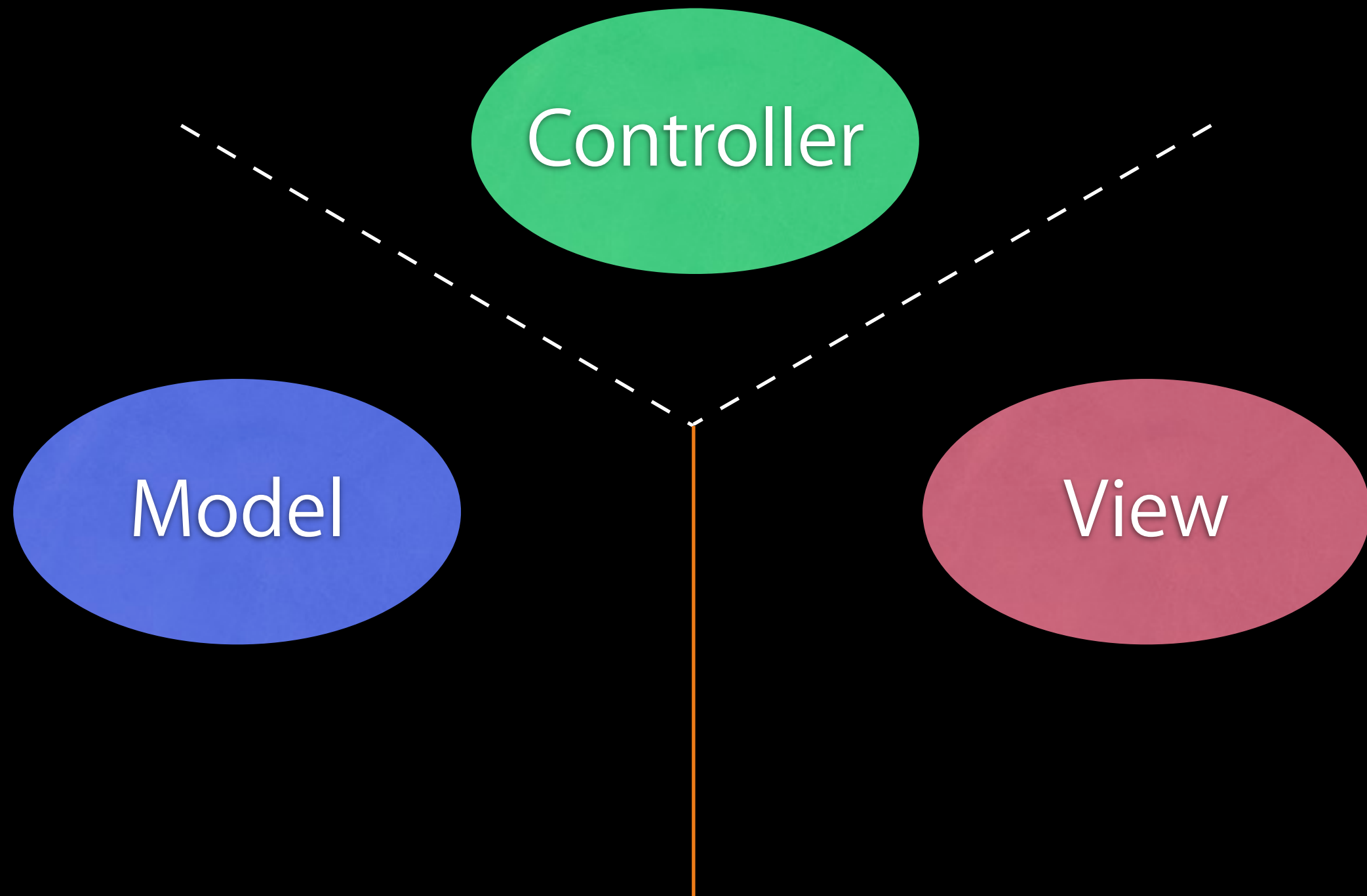
Info.plist file

- Property List (often XML), describing your application
 - Icon appearance
 - Status bar style (default, black, hidden)
 - Orientation
 - Uses Wifi networking
 - System Requirements
- Can edit most properties in Xcode
 - Project > Edit Active Target “Foo” menu item
 - On the properties tab

Model, View, Controller

If you take nothing else away from this class...

Model, View, Controller



Model

- Manages the app data and state
- Not concerned with UI or presentation
- Often persists somewhere
- Same model should be reusable, unchanged in different interfaces

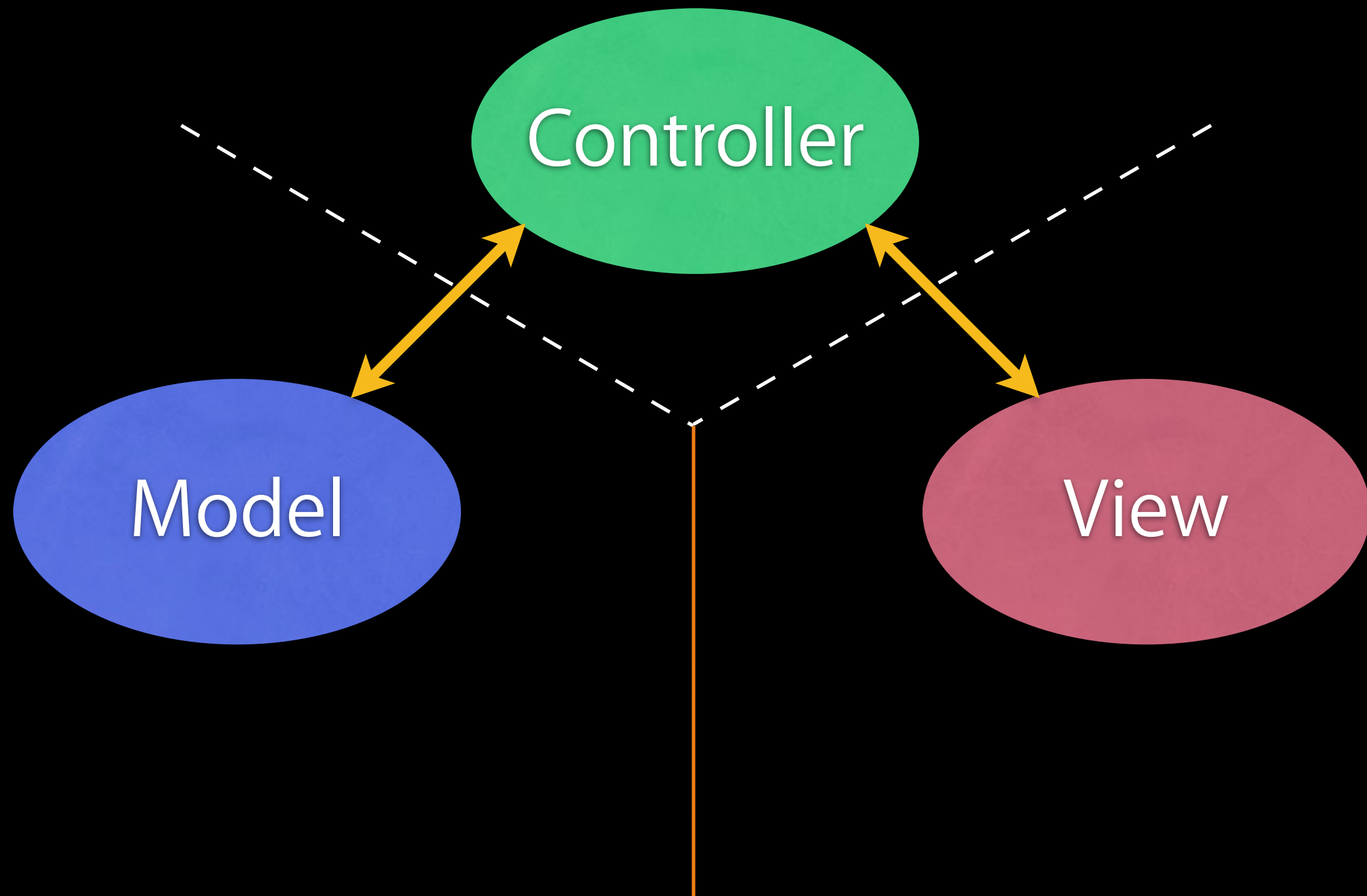
View

- Present the Model to the user in an appropriate interface
- Allows user to manipulate data
- Does not store any data
 - (except to cache state)
- Easily reusable & configurable to display different data

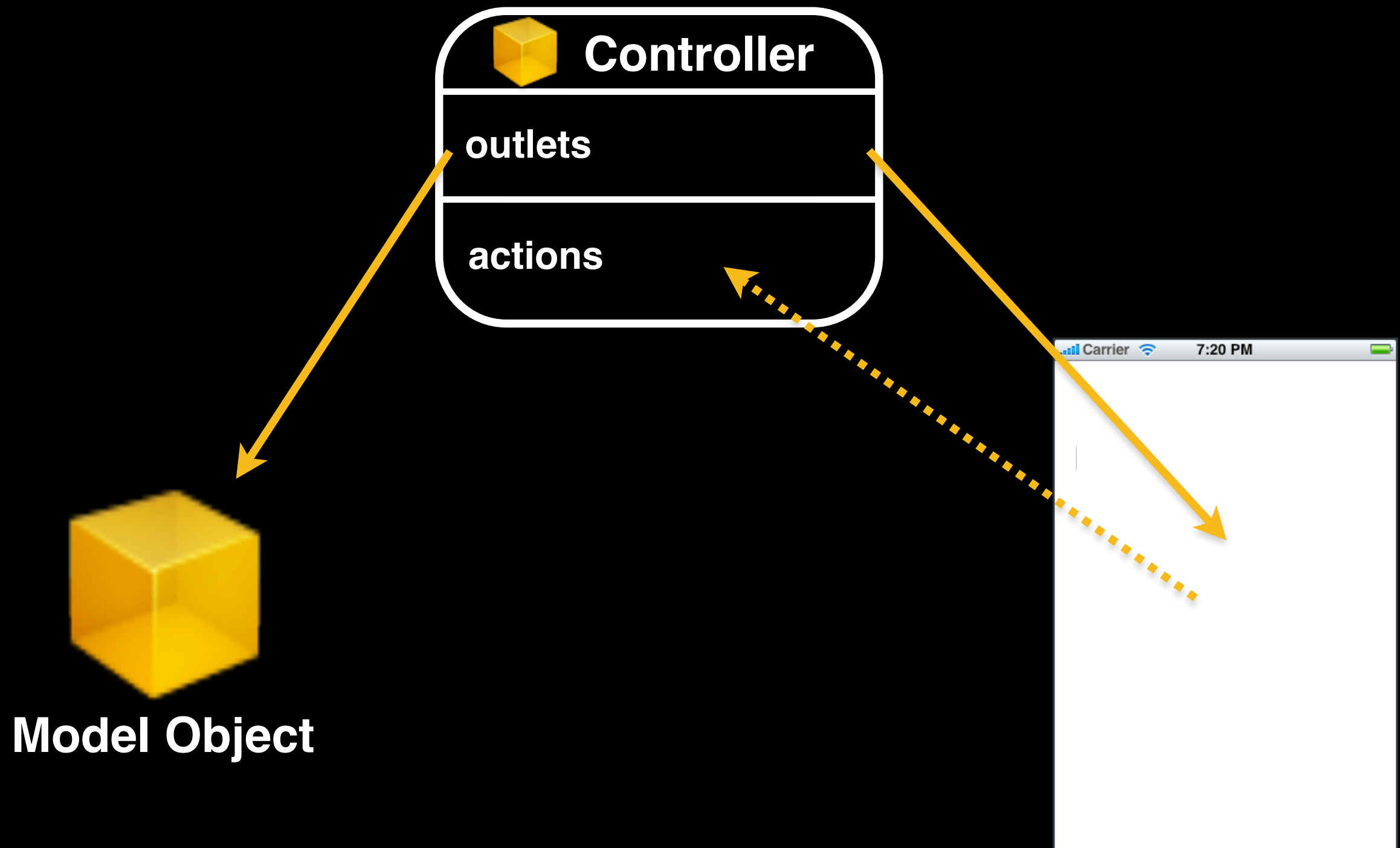
Controller

- Intermediary between Model & View
- Updates the view when the model changes
- Updates the model when the user manipulates the view
- Typically where the app logic lives.

Model, View, Controller

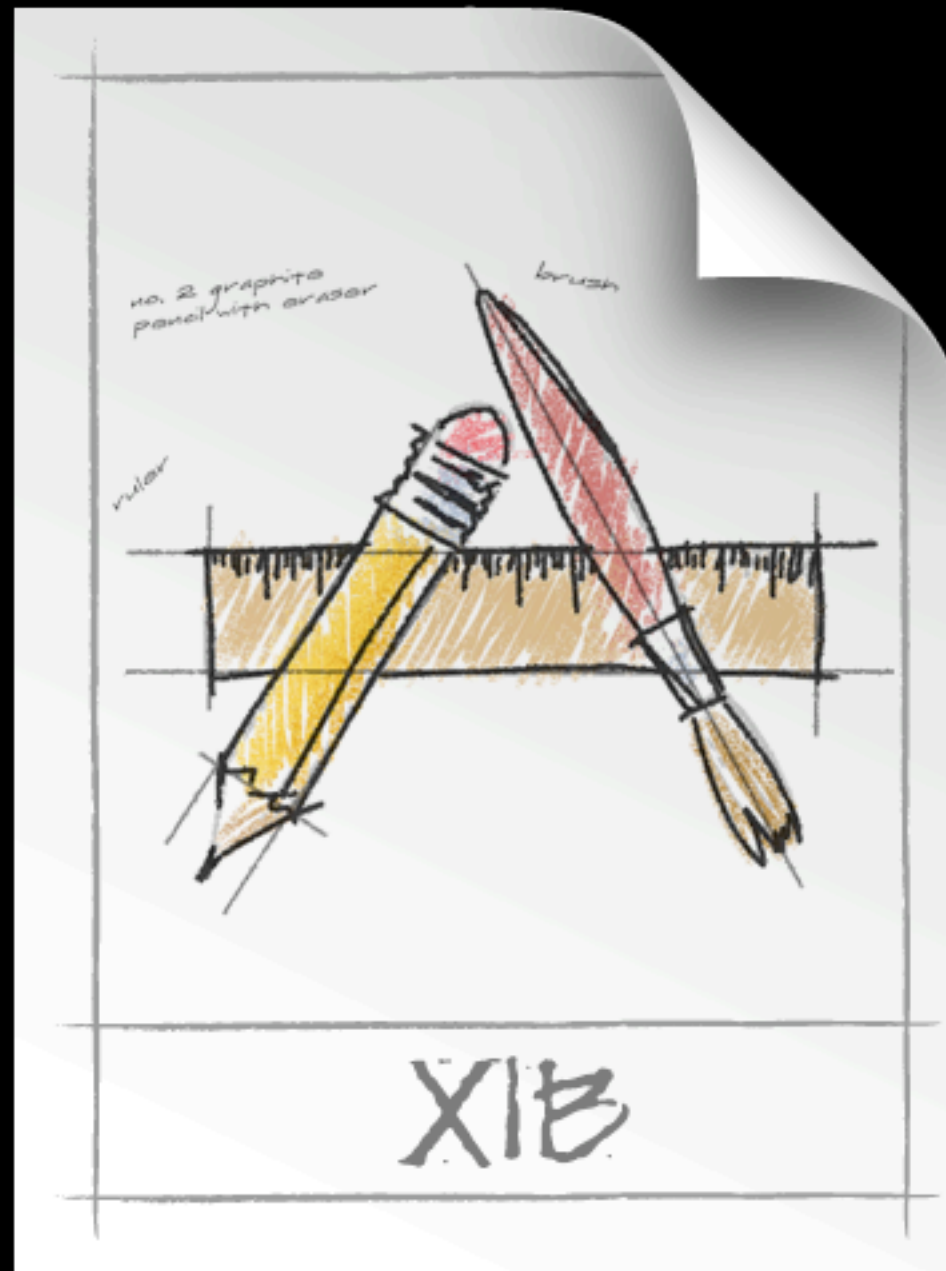


Model, View, Controller



Interface Builder and Nibs

Nib files



Nib Files - Design time

- Helps you design the 'V' in MVC:
 - layout user interface elements
 - add controller objects
 - Connect the controller and UI

Nib Loading

- At runtime, objects are unarchived
 - Values/settings in Interface Builder are restored
 - Ensures all outlets and actions are connected
 - Order of unarchiving is not defined
- If loading the nib automatically creates objects and order is undefined, how do I customize?
 - For example, to displaying initial state

-awakeFromNib

- Control point to implement any additional logic after nib loading
- Default empty implementation on NSObject
- You often implement it in your controller class
 - e.g. to restore previously saved application state
- Guaranteed everything has been unarchived from nib, and all connections are made before -awakeFromNib is called
 - (void)awakeFromNib {
 // do customization here

}

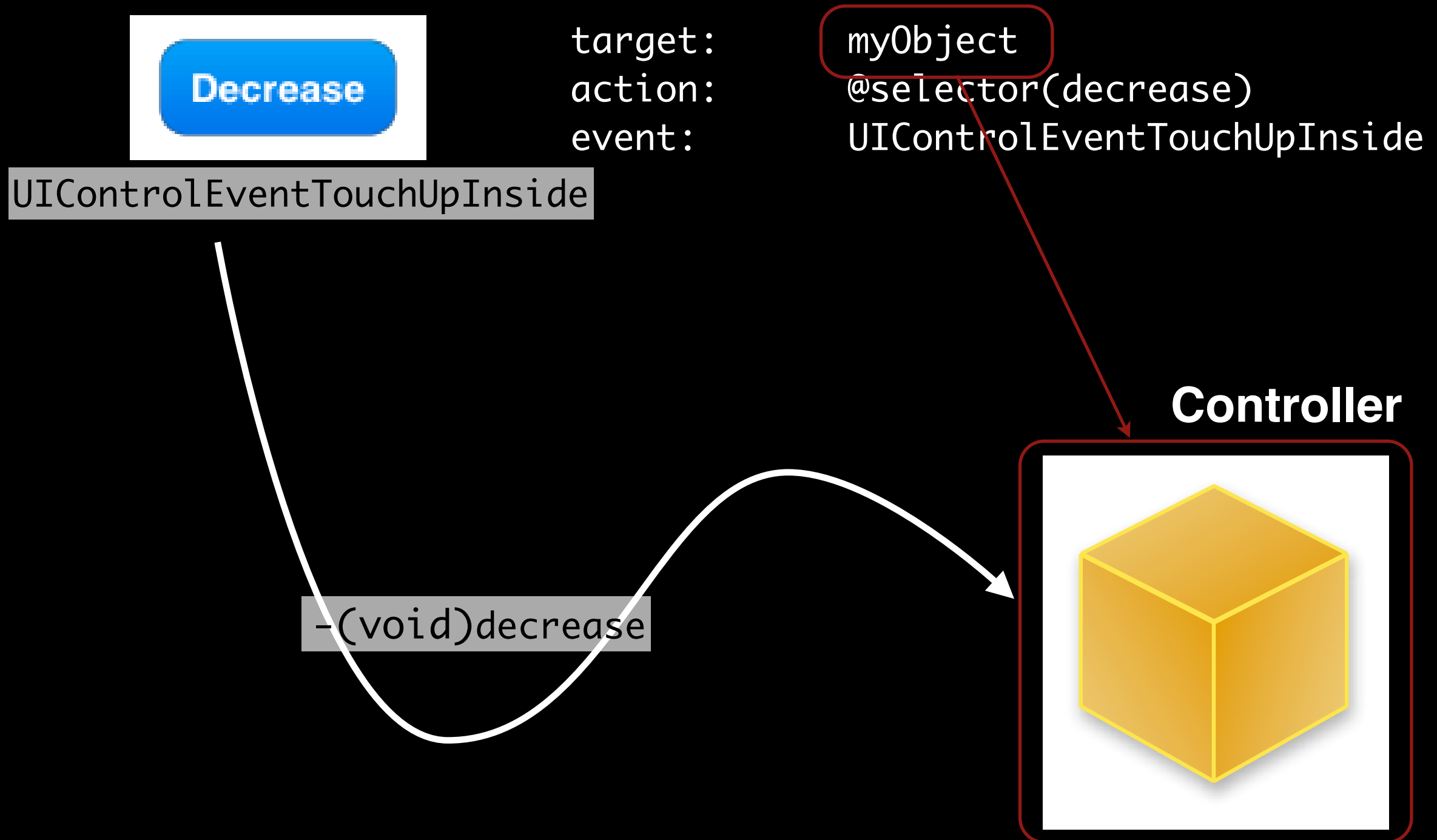
Controls and Target-Action

Controls - Events

- View objects that allows users to initiate some type of action
- Respond to variety of events
 - Touch events
 - touchDown
 - touchDragged (entered, exited, drag inside, drag outside)
 - touchUp (inside, outside)
 - Value changed
 - Editing events
 - editing began
 - editing changed
 - editing ended

Controls - Target/Action

- When event occurs, action is invoked on target object



Action Methods

- 3 different flavors of action method selector types
 - `(void)actionMethod;`
 - `(void)actionMethod:(id)sender;`
 - `(void)actionMethod:(id)sender withEvent:(UIEvent *)event;`
- UIEvent contains details about the event that took place

Action Method Variations

- Simple no-argument selector

```
- (void)increase {  
    // bump the number of sides of the polygon up  
    polygon.numberOfSides += 1;  
}
```

- Single argument selector - control is 'sender'

```
// for example, if control is a slider...  
- (void)adjustNumberOfSides:(id)sender {  
    polygon.numberOfSides = [sender value];  
}
```

Action Method Variations

- Two-arguments in selector (sender & event)
 - `(void)adjustNumberOfSides:(id)sender
 withEvent:(UIEvent *)event`
`{`
`// could inspect event object if you needed to`
`}`

Multiple target-actions

- Controls can trigger multiple actions on different targets in response to the same event
- Different than Cocoa on the desktop where only one target-action is supported
- Different events can be setup in IB

Manual Target-Action

- Same information IB would use
- API and UIControlEvents found in UIControl.h
- UIControlEvents is a bitmask

```
@interface UIControl
```

- (void)**addTarget**:(id)target **action**:(SEL)action
 forControlEvents:(UIControlEvents)controlEvents;
- (void)**removeTarget**:(id)target **action**:(SEL)action
 forControlEvents:(UIControlEvents)controlEvents;

```
@end
```

HelloPoly Demo

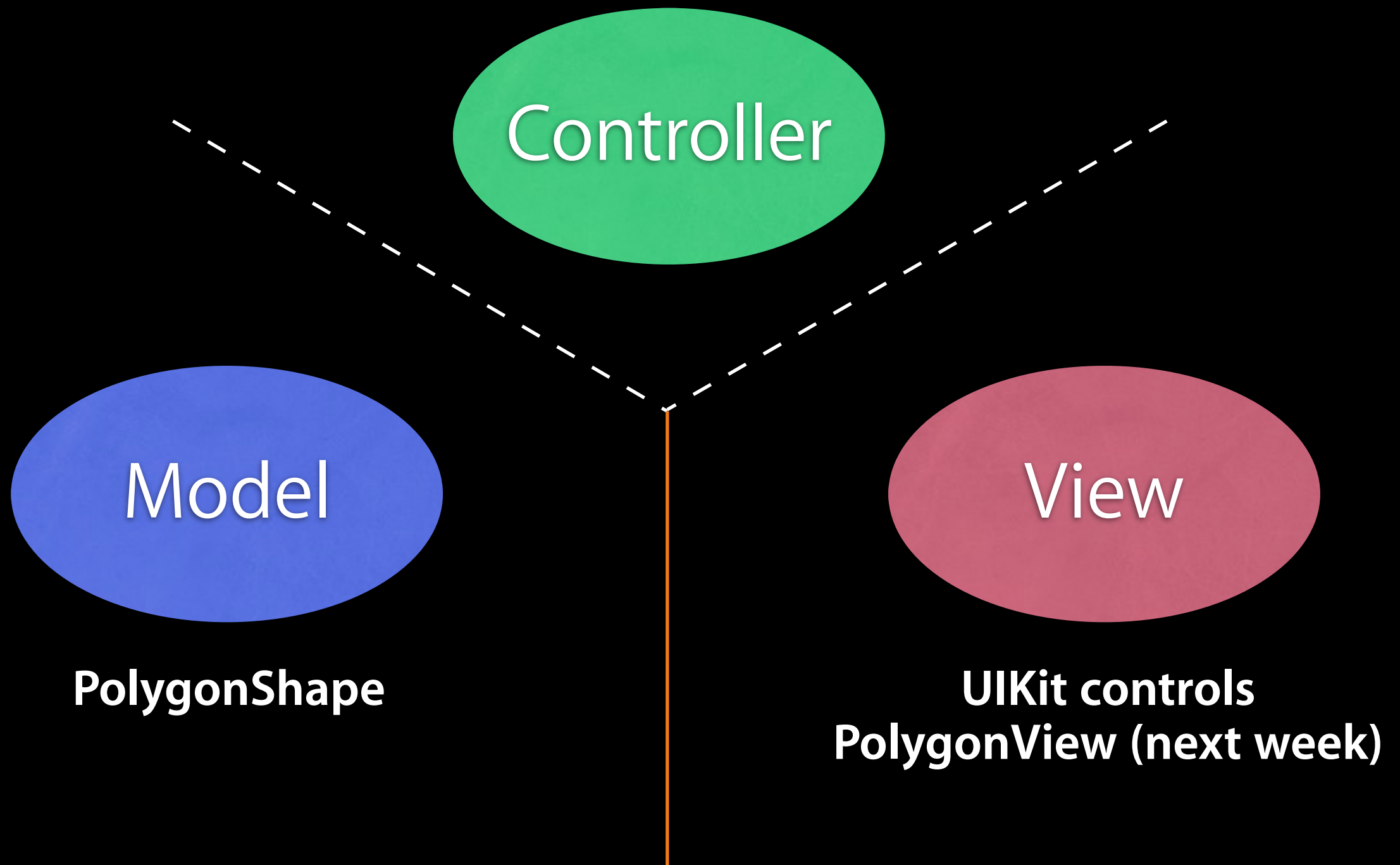
HelloPoly

- This week's assignment is a full MVC application
- Next week's assignment will flesh it out further
- It is not designed to be a complex application
 - rather, provide a series of small studies of the fundamentals of a Cocoa Touch application

Model, View, Controller

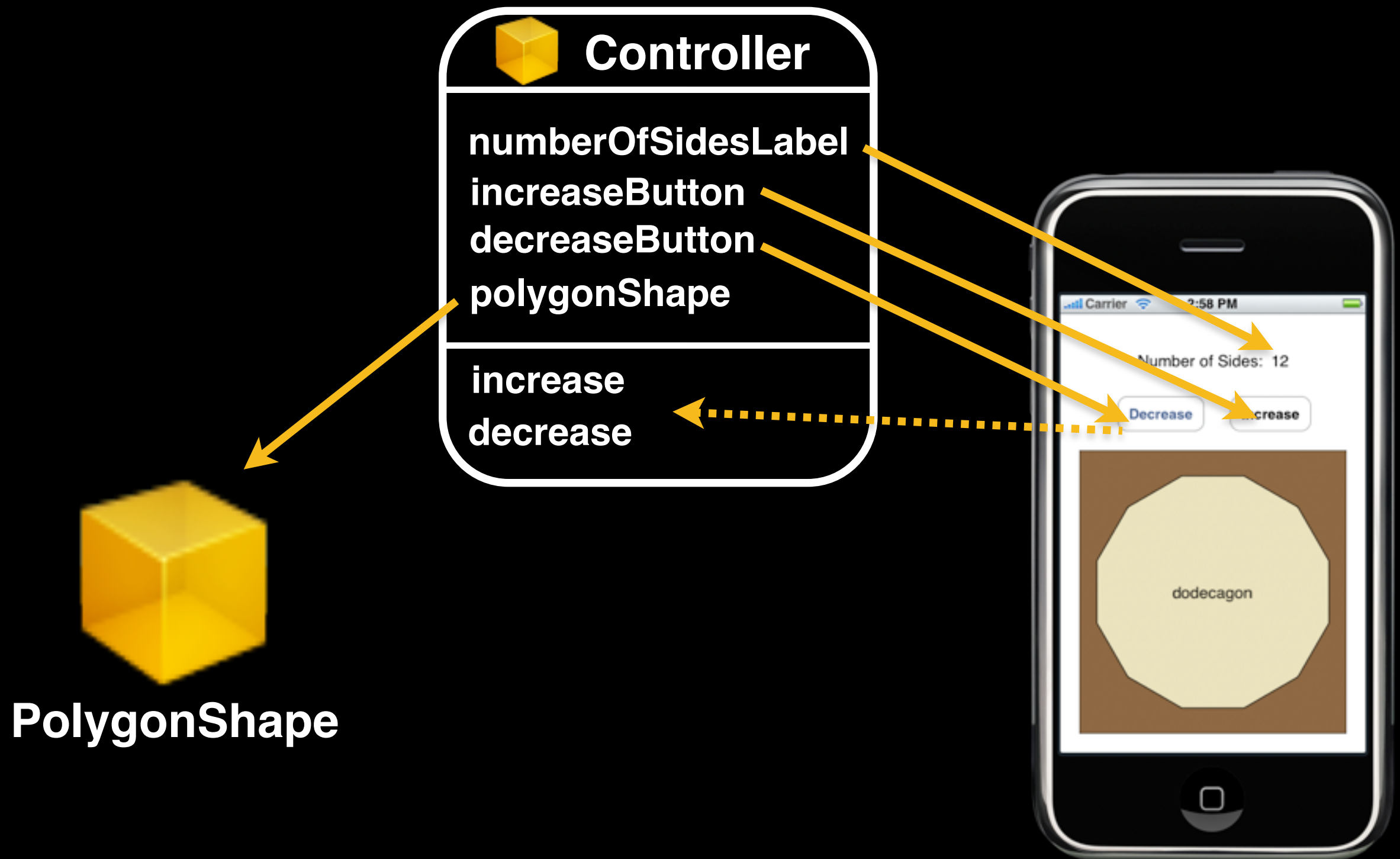
HelloPoly

Controller



Model, View, Controller

HelloPoly



Nib Files - HelloPoly example

- HelloPoly has all objects (model, view and controller) contained in the same MainWindow.xib file
 - More common to have UI broken up into several nib files
- UIKit provides a variety of controllers
 - Evan will be introducing them with the Presence projects

Questions?