Event Handling in iOS

Dong Duong

April 2013

Course Objective

- How to handle event when user interact on the device.
- Using Gesture Recognizers

Prerequisite

- Joined iOS overview course

Assessment Disciplines

- Class Participation : Required
- Assignment Completion : 100%
- Pass Score : >=70%

Course Timetable

- Lecture Duration + Hands-on Labs: 3 hours

Agenda

- Handle basic interaction.
- Demo & Practice
- Gesture Recognizers
- Practice
- Motion Events
- Q&A

Handle basic interaction

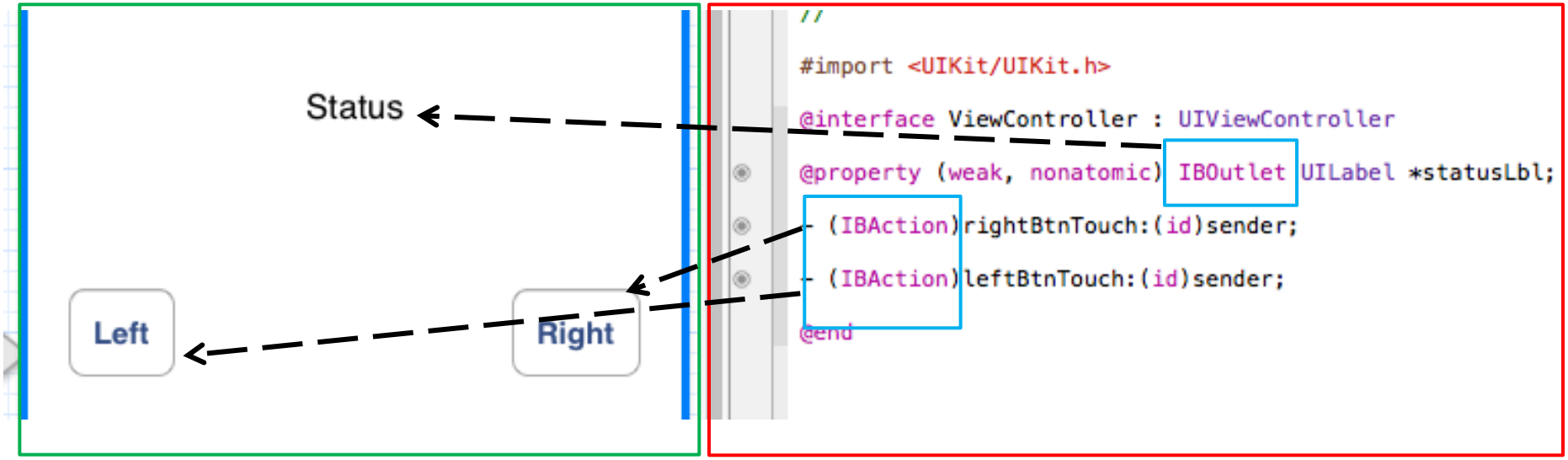
Outlet and Actions

- **Outlet**

  Outlets are special Objective-C class properties that are declared using the keyword IBOutlet

- **Actions**

In a nutshell, actions are methods that are declared with a special return type, IBAction, which tells Interface Builder that this method can be triggered by a control in a nib file.

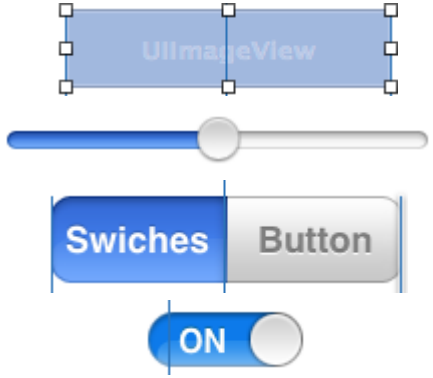


Xib file

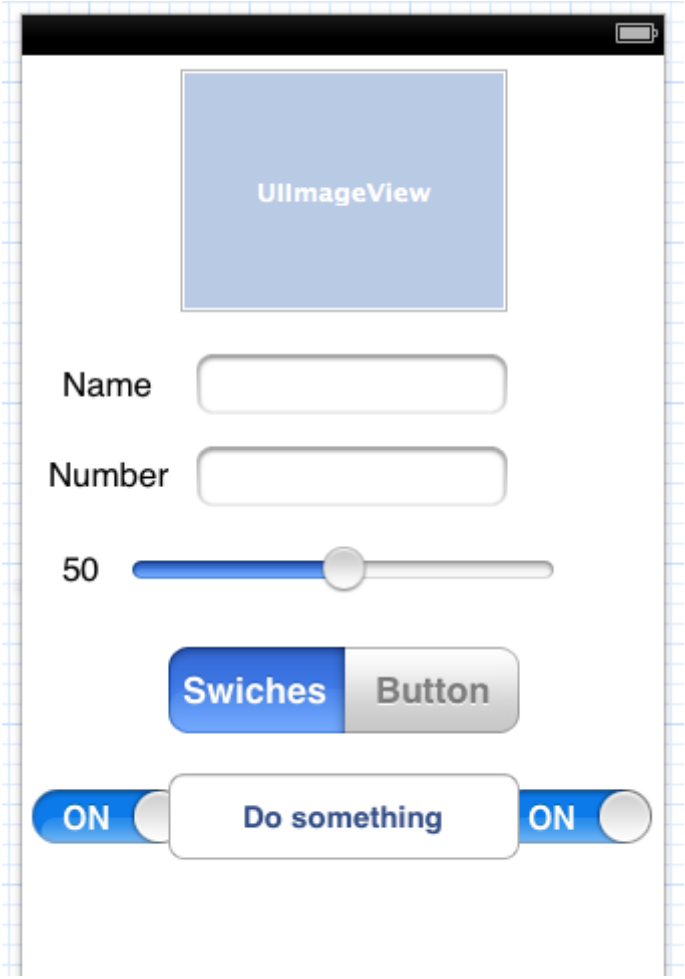
Header file

More User Interface

- Image view
- Slider
- Segmented
- Switches
- Action sheet
- Alert



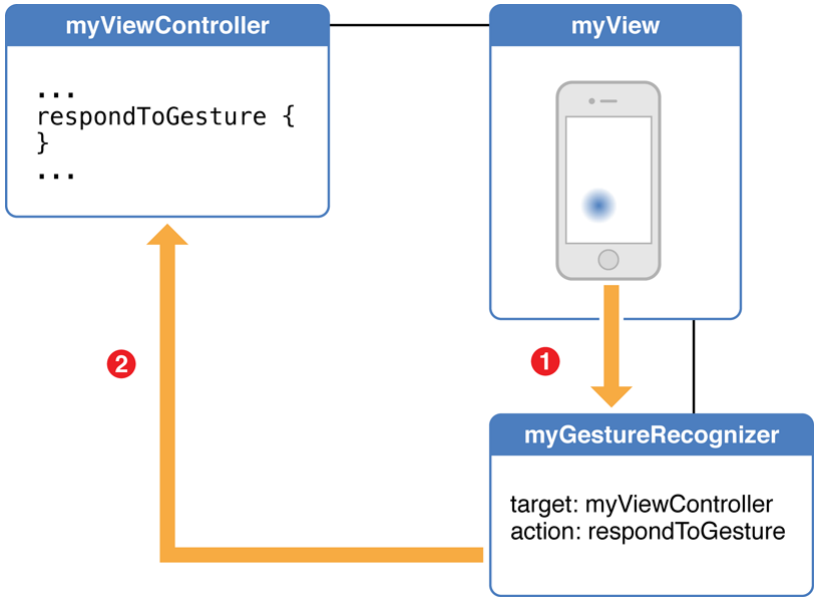
Demo & Practices: Create a application with views as below and Handle action when user interact with controls



Gesture Recognizers

Gesture Recognizers

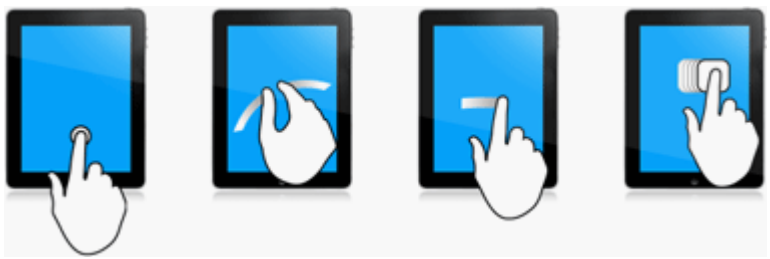
- Gesture recognizers interpret touches to determine whether they correspond to a specific gesture, such as a swipe, pinch, or rotation.



Gesture Recognizers to Simplify Event Handling

Built-in Gesture Recognizers Recognize Common Gestures

- Tapping (any number of taps) - UITapGestureRecognizer
- Pinching in and out (for zooming a view) - UIPinchGestureRecognizer
- Panning or dragging - UIPanGestureRecognizer
- Swiping (in any direction) - UISwipeGestureRecognizer
- Rotating (fingers moving in opposite directions) - UIRotationGestureRecognizer
- Long press (also known as “touch and hold”) - UILongPressGestureRecognizer

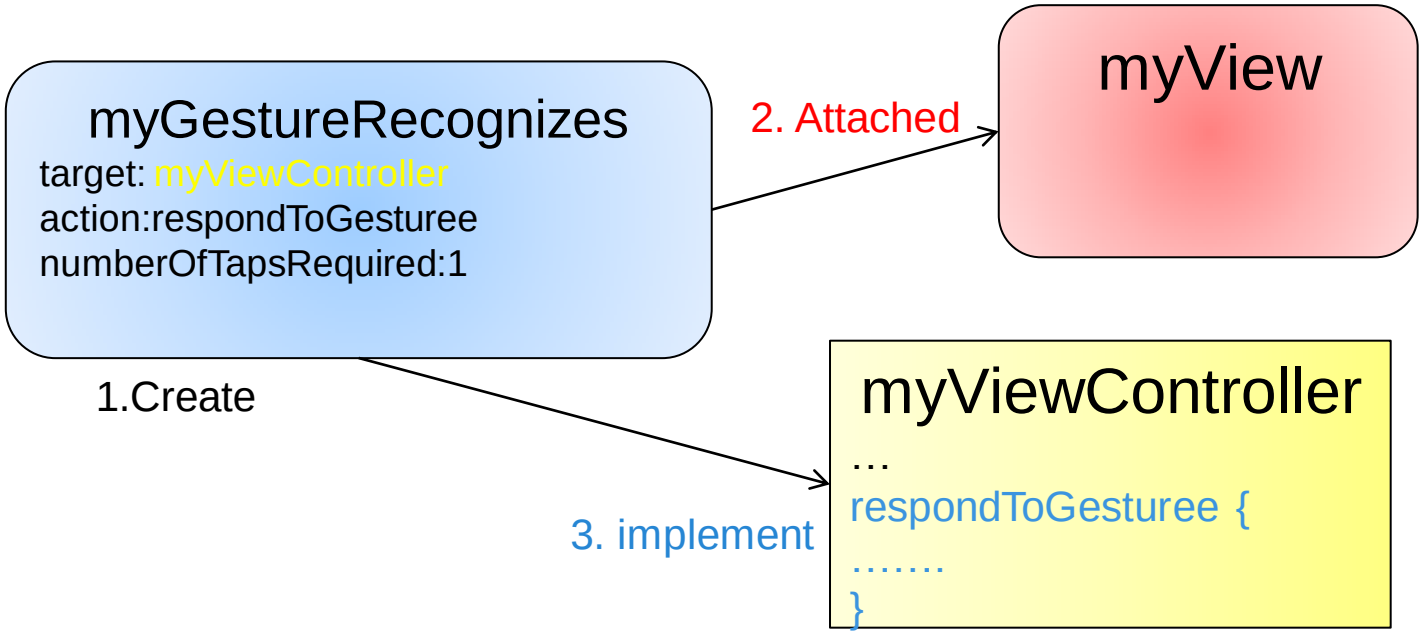


Tap Pinch Swipe Drag

Rotate

Long press

Responding to Events with Gesture Recognizers



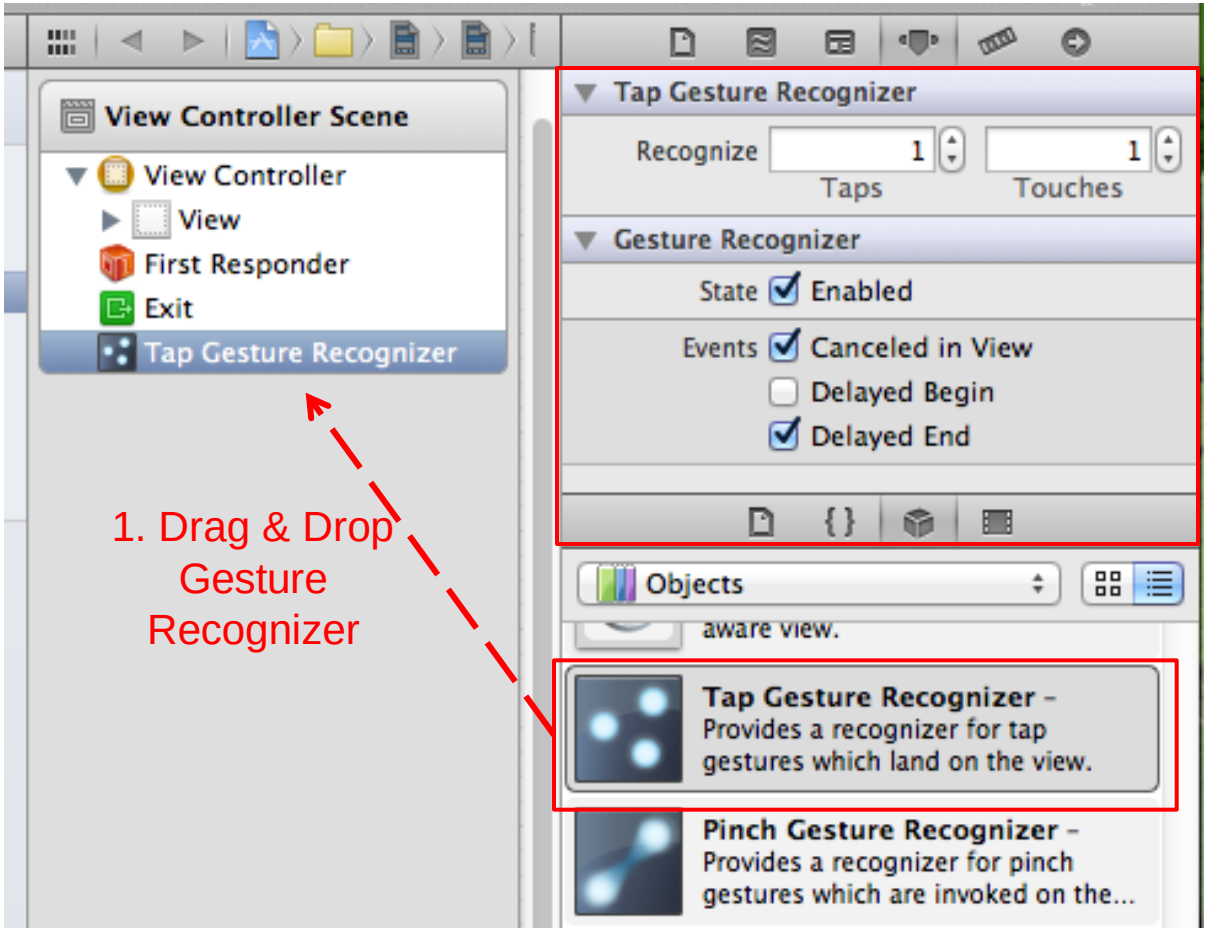
1. Create and configure a gesture recognizer instance.

   This step includes assigning a target, action, and sometimes assigning gesture-specific attributes (such as the numberOfTapsRequired).

2. Attach the gesture recognizer to a view.

3. Implement the action method that handles the gesture.

Create Gesture Recognizers by using Interface Builder



2. Configure gesture recognizer

Create Gesture Recognizers Programmatically

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    // Create and initialize a tap gesture

    UITapGestureRecognizer *tapRecognizer = [[UITapGestureRecognizer alloc]
                initWithTarget:self action:@selector(respondToTapGesture:)];

    // Specify that the gesture must be a single tap
    tapRecognizer.numberOfTapsRequired = 1;

    // Add the tap gesture recognizer to the view
    [self.view addGestureRecognizer:tapRecognizer];
}

- (void)respondToTapGesture:(id)sender {

}
```

1. Create and configure a gesture recognizer instance

Creating a Custom Gesture Recognizer

2. **Import**

```
//

#import <UIKit/UIKit.h>
#import <UIKit/
    UIGestureRecognizerSubclass.h>

@interface CustomGestureRecognizer :
    UIGestureRecognizer

@end
```

1. create a subclass of UIGestureRecognizer

CustomGestureRecognizer.h

```
#import "CustomGestureRecognizer.h"

@implementation CustomGestureRecognizer

- (void)reset {

};

- (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{

};

@end
```

3. Overwrite the methods

CustomGestureRecognizer.m

Practice: Touched on the background to close the keyboard

Motion Event

Responding to changes in device orientation

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // Request to turn on accelerometer and begin receiving accelerometer events
    [[UIDevice currentDevice] beginGeneratingDeviceOrientationNotifications];
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector
        (orientationChanged:) name:UIDeviceOrientationDidChangeNotification object:nil];
}

- (void)orientationChanged:(NSNotification *)notification {

    // Respond to changes in device orientation

}

-(void) viewDidDisappear {

    // Request to stop receiving accelerometer events and turn off accelerometer
    [[NSNotificationCenter defaultCenter] removeObserver:self];
    [[UIDevice currentDevice] endGeneratingDeviceOrientationNotifications];
}
```

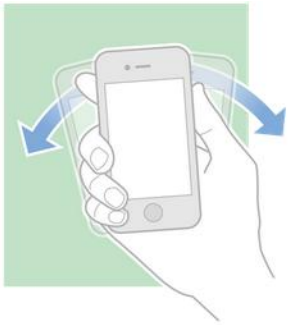
Enabling orientation notifications

Register to receive notification

Stop to receive notification

disable orientation notifications

Detecting Shake-Motion Events with UIEvent



```
- (BOOL)canBecomeFirstResponder {

    return YES;
}

- (void)viewDidAppear:(BOOL)animated {

    [self becomeFirstResponder];
}
```

Designating a First Responder

```
- (void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event
    {

    if (motion == UIEventSubtypeMotionShake)
    {
        // User was shaking the device. Post a notification named
            "shake."
        [[NSNotificationCenter defaultCenter]
            postNotificationName:@"shake" object:self];
    }
}
```

Handling a motion event

THANK YOU


CSC
BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING