

# The First Segment

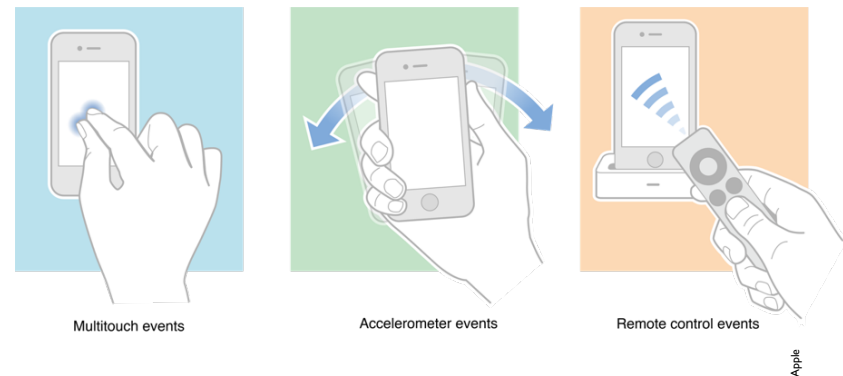
## iPhone Application Programming Lecture 7: Touches & Sensor Input

Nur Al-huda Hamdan  
Christian Corsten  
Media Computing Group  
RWTH Aachen University  
Winter Semester 2013/2014  
<http://hci.rwth-aachen.de/iphone>

- Events
  - UIEvent object, types, responder chain
- Multitouch events
  - UITouch object, phases, response
- Gestures
  - Attach gesture recognizers, state machine, custom gestures

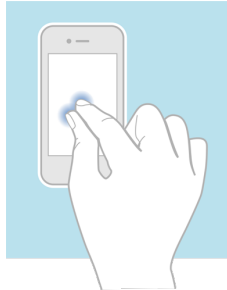
## Events

## Events



# Event Delivery

User-generated event



System



Inside your app



UIApplication

UIWindow



First responder Hit-test view

# UIEvent Types

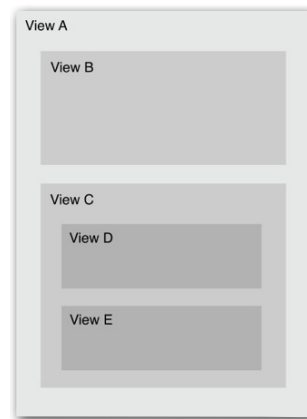
```
typedef enum {
    UIEventTypeTouches,
    UIEventTypeMotion,
    UIEventTypeRemoteControl,
} UIEventType;

typedef enum {
    // available in iPhone OS 3.0
    UIEventSubtypeNone = 0,
    // for UIEventTypeMotion, available in iPhone OS 3.0
    UIEventSubtypeMotionShake = 1,
    // for UIEventTypeRemoteControl, available in iOS 4.0
    UIEventSubtypeRemoteControlPlay = 100,
    UIEventSubtypeRemoteControlPause = 101,
    UIEventSubtypeRemoteControlStop = 102,
    UIEventSubtypeRemoteControlTogglePlayPause = 103,
    UIEventSubtypeRemoteControlNextTrack = 104,
    UIEventSubtypeRemoteControlPreviousTrack = 105,
    UIEventSubtypeRemoteControlBeginSeekingBackward = 106,
    UIEventSubtypeRemoteControlEndSeekingBackward = 107,
    UIEventSubtypeRemoteControlBeginSeekingForward = 108,
    UIEventSubtypeRemoteControlEndSeekingForward = 109,
} UIEventSubtype;
```

UIEvent.h

# Hit-test View

- Hit-test view is the lowest view that contains the touch
- On top most view (A)
  - hitTest:withEvent:
    - pointInside:withEvent:
    - YES: recursively call hitTest:withEvent: on children (subviews)
    - NO: the touch is not in this view or its children, back to super view

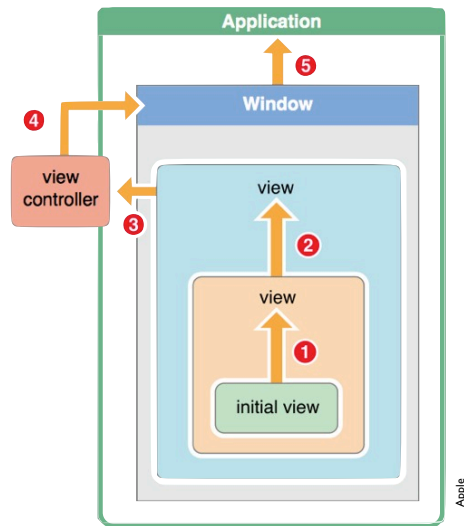


Apple

# The First Responder

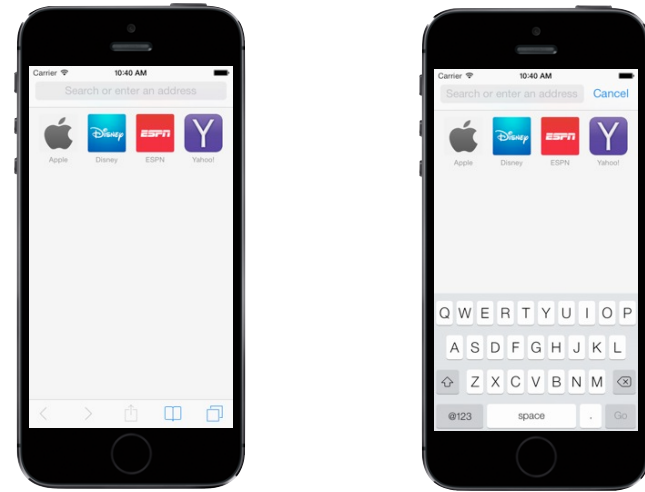
- Designated object to receive events first
- Called from UIWindow directly
- Receives the following events
  - Motion events, Remote-control events, Action messages, Editing-menu messages
- Explicit: override canBecomeFirstResponder method to return YES or receive a becomeFirstResponder message

## Responder Chain



9 iPhone Application Programming • Prof. Jan Borchers

## Input Views



10 iPhone Application Programming • Prof. Jan Borchers

## Handling Text Field Input

```
// UITextField Delegate Method
- (BOOL)textFieldShouldReturn:(UITextField *)textField
{
    // Give feedback if input is invalid,
    // e.g., not a valid email address

    // Give back the first responder status
    [textField resignFirstResponder];
    return YES;
}
```

11 iPhone Application Programming • Prof. Jan Borchers

## Multitouch Events

12 iPhone Application Programming • Prof. Jan Borchers

# Touch

- Each touch is bound to a single finger on the screen
  - *when* and *where* (reduced to a single timestamp and a single point)

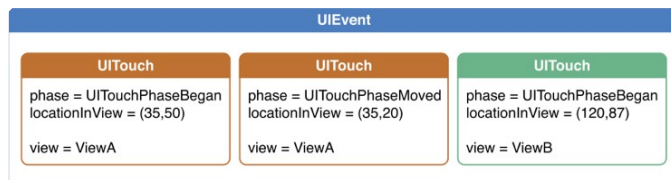


# UITouch

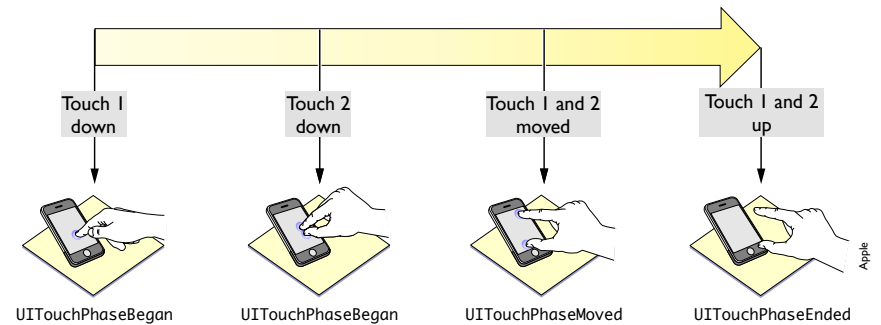
- Represents single touch
- Location can be reported for a given view
- Previous location included
- Additional properties:
  - `tapCount`
  - `timestamp`
  - `phase` (began, moved, stationary, ended, cancelled)
- Attached gesture recognizers

# UITouch in UIEvent

- Stores touches
  - By view (hit-test view) and window
  - For gesture recognizers
- Additional properties:
  - Timestamp
  - Type: touches, motion, or remote-control
  - Subtype: event description for non-touch events



# Touch Phases



# Handling Touch Events

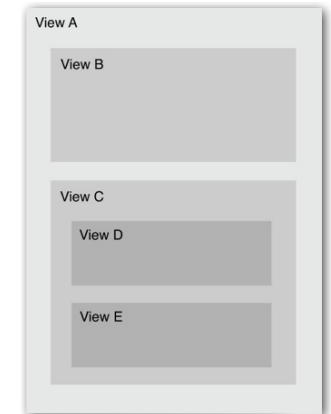
```
// initial touch
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event

// updated touch
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event

// cancelled touch (by external event)
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event

// finished touch
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
```

# Handling Touch Events



# Tracing a UITouch

- UITouch objects don't have an ID, and you cannot retain them in your code because they keep changing!

```
// keep a reference for a touch
for (UITouch *touch in touches) {
    NSValue *key = [NSValue valueForKeyWithPointer:touch];
    [myTouches setValue:FirstFinger forKey:key];
}

// to retrieve a touch
for (UITouch *touch in touches) {
    NSValue *key = [NSValue valueForKeyWithPointer:touch];
    NSObject *valueFromDictionary = [myTouches valueForKey:key];
}
```

# UIControl: Pre-defined Responses

- Subclass of UIView
  - UI elements for control: buttons, sliders, etc.
- Send action messages
- Additional properties:
  - State: enabled, selected, highlighted

## Demo: TouchEvents

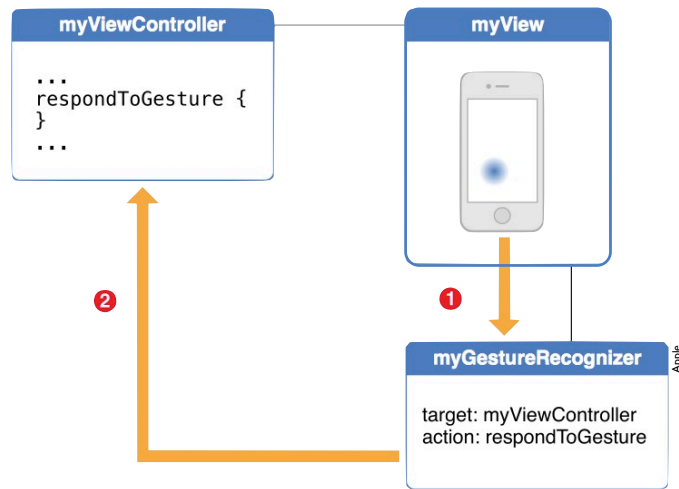
## Demo: DragSubView

## Gesture Recognizers

## Predefined Gesture Recognizers

Gesture	UIKit class
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

# Attaching Gesture Recognizers



# Attaching a Gesture Recognizer

1. Create and initialize a gesture recognizer (in VC)
 

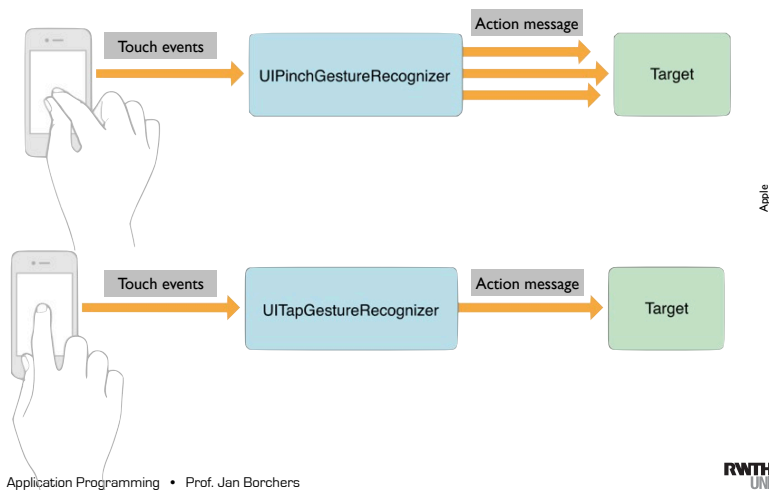
```
UITapGestureRecognizer *tapRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(respondToTapGesture)];
```
2. Configure that gesture
 

```
tapRecognizer.numberOfTapsRequired = 1;
```
3. Add the tap gesture recognizer to the view
 

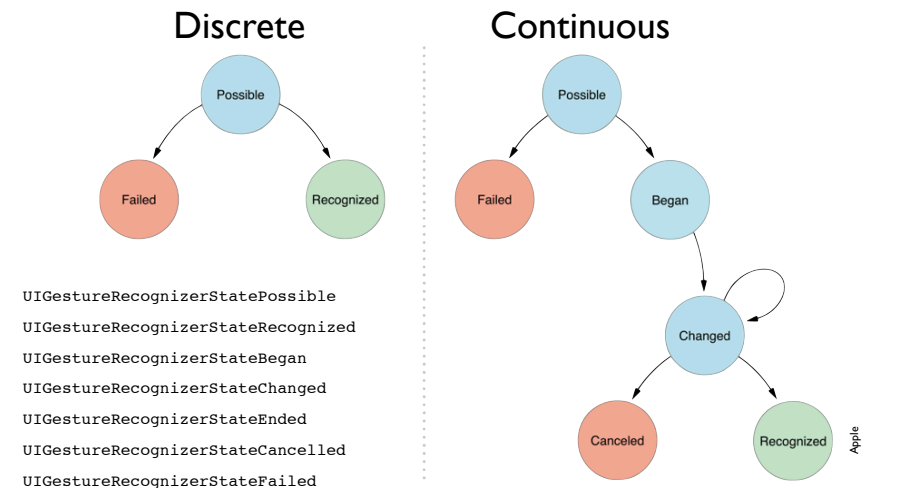
```
[self.view addGestureRecognizer:tapRecognizer];
```
4. Implement the action method that handles the gesture (in V)
 

```
-(void) respondToTapGesture: (UITapGestureRecognizer*)recognizer {...}
```

# Continuous and Discrete Gestures



# State Machines for Gesture Recognizers



# Custom Gesture Recognizers

1. Create a subclass of `UIGestureRecognizer` in Xcode
2. Add to header: `#import <UIKit/UITapGestureRecognizerSubclass.h>`
3. Add to your implementation file:
  - `touchesMoved:withEvent:`
  - `touchesEnded:withEvent:`
  - `touchesCancelled:withEvent:`
  - `touchesBegan:withEvent:`
4. Reset internal state  
`reset`
5. Avoid conflicting gestures
  - `canBePreventedByGestureRecognizer:`
  - `canPreventGestureRecognizer:`

## Demo: GestureRecognizer

## Core Motion

## Motion Events

- Much simpler than using sensor data
- Only a shake-motion is defined
- Usage
  - Make your view first responder
  - Implement the following methods
    - `(void)motionBegan:(UIEventSubtype)motion withEvent:(UIEvent *)event`
    - `(void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event`
    - `(void)motionCancelled:(UIEventSubtype)motion withEvent:(UIEvent *)event`
- `ApplicationSupportsShakeToEdit`



# Device Orientation

- Tell `UIDevice` to generate device orientation notifications

`beginGeneratingDeviceOrientationNotifications`

- Register to receive these notification

`UIDeviceOrientationDidChangeNotification`

- Turn off device orientation notifications

`endGeneratingDeviceOrientationNotifications`

# UIAccelerometer

- Alternative to Core Motion

- Only for acceleration

- Usage:

- Get shared instance (singleton)
- Configure update frequency
- Assign delegate
- Acceleration reported as `UIAcceleration`

Objects are updated for performance reasons

# UIAccelerometer

```
- (void)viewWillAppear:(BOOL)animated
{
    UIAccelerometer *a = [UIAccelerometer sharedAccelerometer];
    a.updateInterval = 0.1;
    a.delegate = self;
}

- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:
    (UIAcceleration *)acceleration
{
    NSLog(@"%f %f %f", acceleration.x, acceleration.y, acceleration.z);
}
```

# Accelerometer Update Frequency

10–20	Orientation detection
30–60	Real-time input (e.g., games)
70–100	high-frequency motion (e.g., hitting or shaking the device quickly)

# Accelerometer vs. Gyroscope

- Accelerometer
  - Measures proper acceleration
  - Relative to free fall
  - 1.0 = 1G (earth's acceleration)
- Gyroscope
  - Measure rotation

# Accelerometer vs. Gyroscope



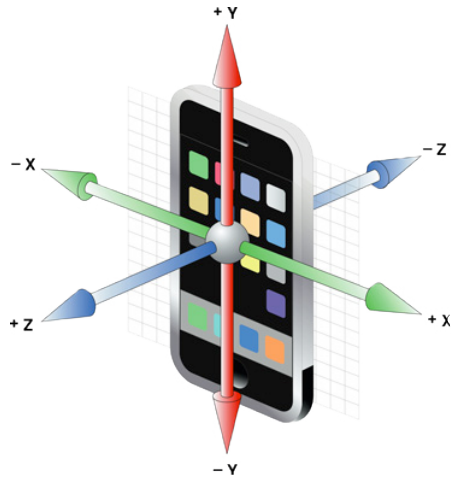
# Core Motion

- Obtain motion data from available sensors
  - Accelerometer (alternative to `UIAccelerometer`)
  - Gyroscope
- Framework
  - `CMMotionManager`
  - `CMAccelerometerData`
  - `CMGyroData`
  - `CMDeviceMotion`

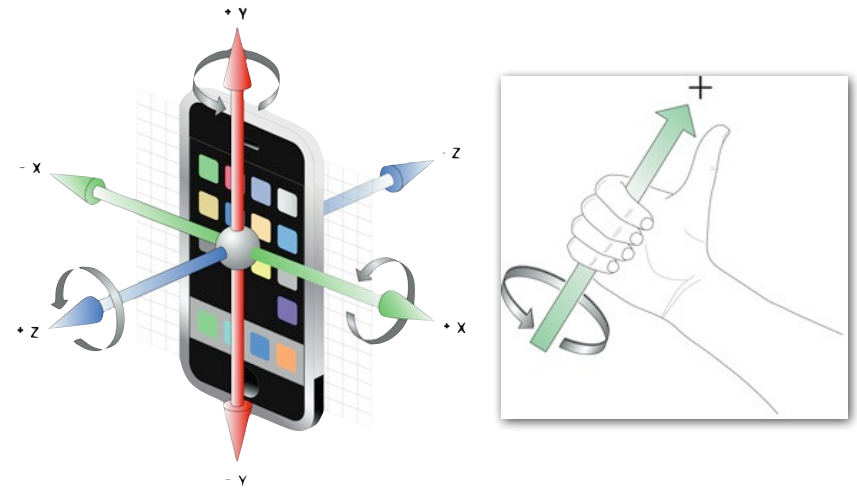
# CMMotionManager

- Operates on accelerometer, gyro, or both
- Updating with handler:
  - `startXUpdates`
  - `startXUpdatesToQueue:withHandler:`
  - Block is added to `NSOperationQueue`
- Updating without handler:
  - `startXUpdates`
  - Query sensor data when needed (e.g., through timer)
- `X = [Accelerometer | Gyro | DeviceMotion]`

## CMAcceleration

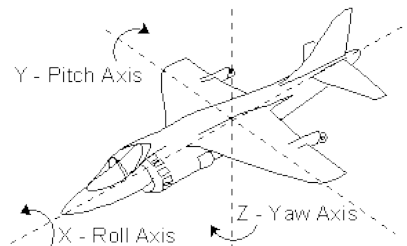


## CMGyroData



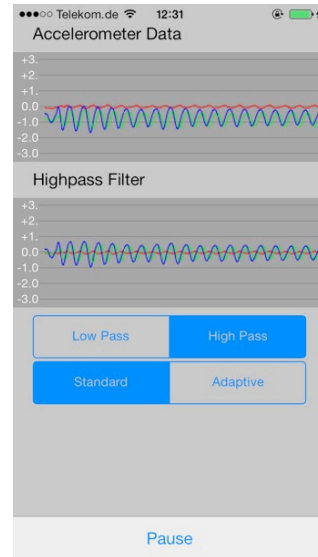
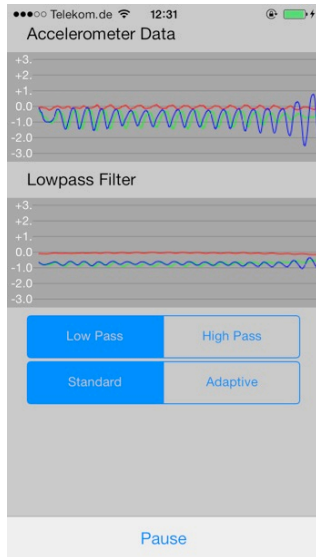
## CMDeviceMotion

- Only available with Gyroscope
- Position in 3D Space
  - Attitude: roll, pitch, yaw, or rotationMatrix, or quaternion
  - x, y, z rotation
- Acceleration
  - Gravity vector
  - User acceleration vector



## Filtering Data

- Low-pass filter
  - Pass low-frequency, cut off high-frequency signals
  - Detect orientation changes
  - Reduces jittering
- High-pass filter
  - Pass high-frequency, cut off low-frequency signals
  - Detect jittering
  - Returns relative value



# Low-Pass / High-Pass Filter

```

// low-pass filter
CGFloat lowpassFilter(CGFloat value, CGFloat filterFactor) {
    static CGFloat lowpassValue;
    lowpassValue = value*filterFactor + lowpassValue*
        (1.0 - filterFactor);
    return lowpassValue;
}

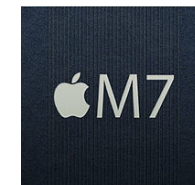
// high-pass filter
CGFloat highpassFilter(CGFloat value, CGFloat filterFactor) {
    static CGFloat prevValue, highpassValue;
    highpassValue = filterFactor * (highpassValue+value-
        prevValue);
    prevValue = value;
    return highpassValue;
}

```

## Demo: Marble

## iOS7: M7 Coprocessor

- Only for iPhone 5S, iPad Air, and iPad mini with Retina display
- Accelerometer, gyroscope, compass
- Measures motion data continuously without running down the battery
- Used for step counting, fitness/health apps
- Check Core Motion Framework Reference



# New Classes for M7

- Use `CMMotionActivityManager` to start/stop activity updates
- Updates are delivered as instances of `CMMotionActivity` objects
- A `CMMotionActivity` object contains all data for each motion event
  - Boolean properties: `stationary`, `running`, `walking`, `automotive`
  - Other properties: `startDate`, `confidence`

# New Classes for M7

- `CMStepCounter`: record the user's steps
  - Use `isStepCountingAvailable` method to check whether device supports step counting (`YES`) or not (`NO`)
- Start listening for steps:
  - `(void)startStepCountingUpdatesToQueue:(NSOperationQueue *)queue  
updateOn:(NSInteger)stepCounts  
withHandler:(CMStepUpdateHandler)handler;`
- `updateOn:(NSInteger)stepCounts` to determine after how many steps your app should be notified about step updates
- M7 records steps even if the app is not asking for them

## Demo: Motion Activity & Step Counting

## Other Input

# Proximity Sensor

- Located at the top of the phone
- Triggered at a distance of ~5cm
- Default behavior (phone app):
  - Turn off display / touch sensing



# Using the Proximity Sensor

```
// enable proximity monitoring
[[UIDevice currentDevice] setProximityMonitoringEnabled:YES];

// register for notifications
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(proximityChanged:)
 name:UIDeviceProximityStateDidChangeNotification
 object:[UIDevice currentDevice]];

// handle proximity change
- (void)proximityChanged:(NSNotification *)notification {
    BOOL proximityState = [[notification object] proximityState];
    NSLog(@"Proximity Changed: %@", proximityState);
}
```

# Remote-Control

- Become first responder
  - Turn on remote-control events
- ```
[[UIApplication sharedApplication]
beginReceivingRemoteControlEvents];
```
- Implement
- ```
- (void) remoteControlReceivedWithEvent:
(UIEvent *) receivedEvent
```
- Turn off remote-control events
- ```
[[UIApplication sharedApplication]
endReceivingRemoteControlEvents];
```




# Remote-Control

```
- (void) viewWillAppear:(BOOL) animated {
    [super viewWillAppear:animated];
    [[UIApplication sharedApplication] beginReceivingRemoteControlEvents];
    [self becomeFirstResponder];
}

- (void) remoteControlReceivedWithEvent: (UIEvent *) receivedEvent {
    if (receivedEvent.type == UIEventTypeRemoteControl) {
        switch (receivedEvent.subtype) {
            case UIEventSubtypeRemoteControlTogglePlayPause:
                [self playOrStop: nil];
                break;
            case UIEventSubtypeRemoteControlPreviousTrack:
                [self previousTrack: nil];
                break;
            case UIEventSubtypeRemoteControlNextTrack:
                [self nextTrack: nil];
                break;
            default: break;
        }
    }
}}
```

# Summary

- Touch & gesture recognizers
- Core Motion
  - Accelerometer
  - Gyroscope
  - Device motion
  - M7 coprocessor
- Other: proximity, remote-control
- Reading assignment

 Event Handling Guide

