Mobile Application Projects

CSE40333/CSE60333

Spring 2013

Sensor Programming iOS

Lecture 6B

http://www3.nd.edu/~cse/2013sp/60333/

Salvador Aguiñaga

Mobile Application Projects

Lecture Outline

- [ ] Intro to Sensors on iOS Devices
- [ ] iOS Technology: Device Frameworks
- [ ] Examples: Proximity Sensor, Bluetooth, Accelerometer
- [ ] Overview of Core Motion
- [ ] Accelerometer in Depth
- [ ] Sensor Resources

Lecture Goals

- Get an introduction to the hot topic of programming iPhone's built-in sensors
- Learn how to create sensor-aware apps that respond to a user's location
- Understand the basics of augmented reality programming
- Build apps that combine data from the accelerometer, GPS, digital compass, and camera

Sensors on iOS Devices

Cellular & Wireless

- GSM model A1428*: UMTS/HSPA+/DC-HSDPA (850, 900, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz); LTE (Bands 4 and 17)
- CDMA model A1429*: CDMA EV-DO Rev. A and Rev. B (800, 1900, 2100 MHz); UMTS/HSPA+/DC-HSDPA (850, 900, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz); LTE (Bands 1, 3, 5, 13, 25)
- GSM model A1429*: UMTS/HSPA+/DC-HSDPA (850, 900, 1900, 2100 MHz); GSM/EDGE (850, 900, 1800, 1900 MHz); LTE (Bands 1, 3, 5)
- 802.11a/b/g/n Wi-Fi (802.11n 2.4GHz and 5GHz)
- Bluetooth 4.0 wireless technology

Sensors on iOS Devices

Location	Sensors	
Assisted GPS and GLONASS	Three-axis gyro	Built-in microphones
Digital compass	Accelerometer	8-megapixel iSight camera
Wi-Fi	Proximity sensor	
Cellular	Ambient light sensor	

iOS Technology: Device Frameworks

Name	First available	Prefixes	Description
AudioToolbox.framework	2.0	AU, Audio	Contains the interfaces for handling audio stream data and for playing and recording audio. See "Core Audio."
AudioUnit.framework	2.0	AU, Audio	Contains the interfaces for loading and using audio units. See "Core Audio."
AVFoundation.framework	2.2	AV	Contains Objective-C interfaces for playing and recording audio and video. See "AV Foundation Framework."
CFNetwork.framework	2.0	CF	Contains interfaces for accessing the network via Wi-Fi and cellular radios. See "CFNetwork Framework."
CoreAudio.framework	2.0	Audio	Provides the data types used throughout Core Audio. See "Core Audio."
CoreBluetooth.framework	5.0	CB	Provides access to low-power Bluetooth hardware. See "Core Bluetooth Framework"

Proximity Sensor

Using the Proximity Sensor

1. `proximityMonitoringEnabled` property
2. `proximityState` property

proximityState

A Boolean value indicating whether the proximity sensor is close to the user (`YES`) or not (`NO`). (read-only)

```
@property(nonatomic, readonly)
BOOL proximityState
```

Availability

1. Available in iOS 3.0 and later.

See Also

1. `@property proximityMonitoringEnabled`

Declared In

`UIDevice.h`

proximityMonitoringEnabled

A Boolean value indicating whether proximity monitoring is enabled (`YES`) or not (`NO`).

```
@property(nonatomic,
getter=isProximityMonitoringEnabled)
BOOL proximityMonitoringEnabled
```

Discussion

Enable proximity monitoring only when your application needs to be notified of changes to the proximity state. Otherwise, disable proximity monitoring. The default value is `NO`.

Not all iOS devices have proximity sensors.

Bluetooth

Core Bluetooth Framework

The Core Bluetooth framework (`CoreBluetooth.framework`) allows developers to interact specifically with Bluetooth Low-Energy ("LE") accessories. The Objective-C interfaces of this framework allow you to scan for LE accessories, connect and disconnect to ones you find, read and write attributes within a service, register for service and attribute change notifications, and much more.

For more information about the interfaces of the Core Bluetooth framework, see Core Bluetooth Framework Reference.

Core Motion Framework Reference

- Lets your application receive **motion data from device hardware** and process that data.
- This hardware includes an accelerometer and, on some device models, a magnetometer and a gyroscope.
- Through the CMMotionManager class you can start receiving accelerometer, gyroscope, magnetometer, and combined device-motion events at regular intervals or you can poll for them periodically.

Core Motion Framework Reference

Class References

1. CMAccelerometerData
2. CMAttitude
3. CMDeviceMotion
4. CMGyroData
5. CMLogItem
6. CMMagnetometerData
7. CMMotionManager

An instance of this class represents a measurement of the device's attitude at a point in time. "Attitude" refers to the orientation of a body relative to a given frame of reference.

The CMAttitude class offers three different mathematical representations of attitude: a rotation matrix, a quaternion, and Euler angles (roll, pitch, and yaw values).

Encapsulates measurements of the attitude, rotation rate, and acceleration of a device.

An instance of this class contains a single measurement of the device's rotation rate.

An application receives or samples CMGyroData objects at regular intervals after calling the startGyroUpdatesToQueue:withHandler: method or the startGyroUpdates method of the CMMotionManager class.

Sensors on iOS Devices

Location	Sensors	
Assisted GPS and GLONASS	Three-axis gyro	Built-in microphones
Digital compass	**Accelerometer**	8-megapixel iSight camera
Wi-Fi	Proximity Sensor	
Cellular	Ambient light sensor	

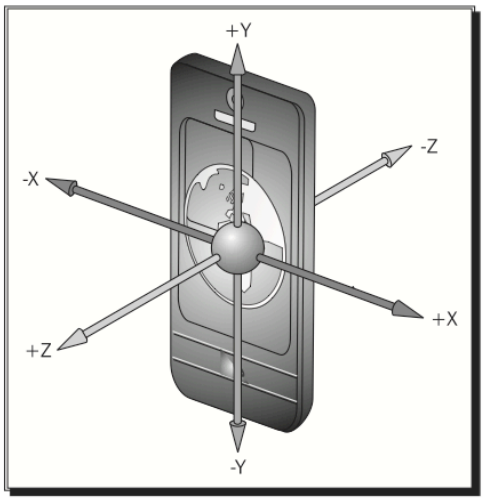

Image Reference {click}

Accelerometer in Depth

UIAccelerometer Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 2.0 and later.

1. Lets you register to receive acceleration-related data from the onboard hardware.
2. You do not create accelerometer objects directly.
    1. Instead, **you use the shared UIAccelerometer object** to specify the interval at which you want to receive events and then set its delegate property.
    2. Upon assigning your delegate object, the accelerometer object begins delivering acceleration events to your delegate immediately at the specified interval. Events are always delivered on the main thread of your application.

Using the UIAccelerometer Class

MainViewController.h

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

@class GraphView;
@class AccelerometerFilter;

@interface MainViewController : UIViewController<UIAccelerometerDelegate>
{
    GraphView *unfiltered;
    GraphView *filtered;
    UIBarButtonItem *pause;
    UILabel *filterLabel;
    AccelerometerFilter *filter;
    BOOL isPaused, useAdaptive;
}
```

MainViewController.m

```
// UIAccelerometerDelegate method, called when the device accelerates.
-(void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:
(UIAcceleration *)acceleration
{
    // Update the accelerometer graph view
    if(!isPaused)
    {
        [filter addAcceleration:acceleration];
        [unfiltered addX:acceleration.x y:acceleration.y z:acceleration.z];
        [filtered addX:filter.x y:filter.y z:filter.z];
    }
}
```

Mobile App Projects

MainViewController.m

```
// Implement viewDidLoad to do additional setup after loading the view.
-(void)viewDidLoad
{
    [super viewDidLoad];
    pause.possibleTitles = [NSSet setWithObjects:kLocalizedPause, kLocalizedResume, nil];
    isPaused = NO;
    useAdaptive = NO;
    [self changeFilter:[LowpassFilter class]];

    [[UIAccelerometer sharedAccelerometer] setUpdateInterval:1.0 / kUpdateFrequency];
    [[UIAccelerometer sharedAccelerometer] setDelegate:self];

    [unfiltered setIsAccessibilityElement:YES];
    [unfiltered setAccessibilityLabel:NSLocalizedString(@"unfilteredGraph", @"")];

    [filtered setIsAccessibilityElement:YES];
    [filtered setAccessibilityLabel:NSLocalizedString(@"filteredGraph", @"")];
}
```

Sensor Resources

Topics Covered:

- AccelerometerGraph
- Core Bluetooth
- UIAcceleration
- CoreMotion Reference
- Proximity Sensor Features

Others:

Sheng, Xiang; Xiao, Xuejie; Tang, Jian; Xue, Guoliang; , "Sensing as a service: A cloud computing system for mobile phone sensing," Sensors, 2012 IEEE , vol., no., pp.1-4, 28-31 Oct. 2012

Sensors and Sensibility - IEEE Spectrum
spectrum.ieee.org/computing/networks/sensors-and-sensibility

Compass Band