Understanding Core Motion

Session 524

Andy Pham
iOS Engineer

Agenda

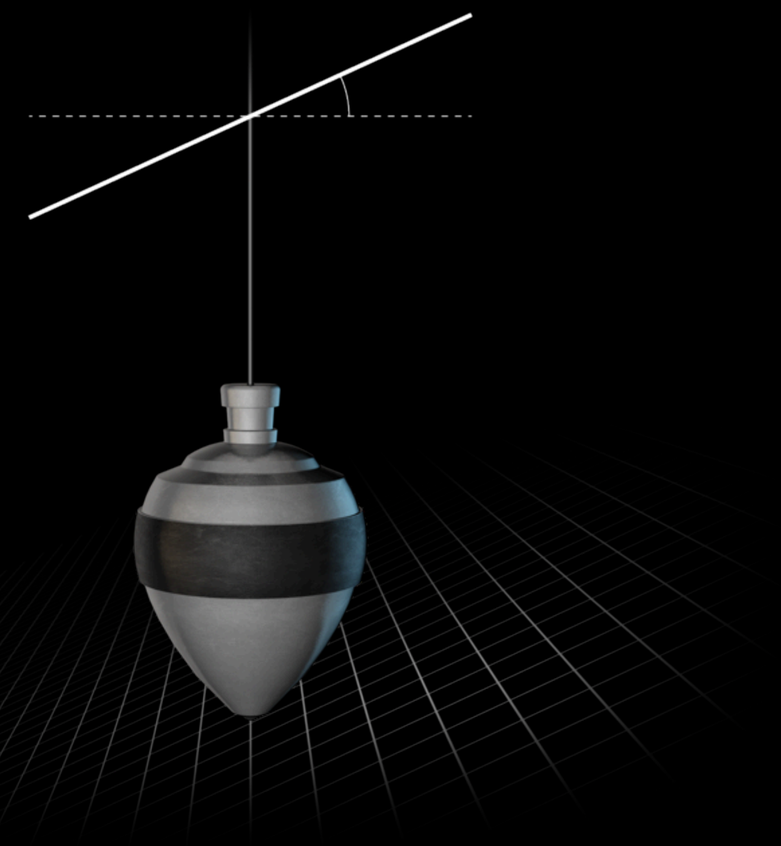
1. What is Core Motion?
2. Sensor fusion deep dive
3. How can I use Core Motion?
4. Let's code!

Core Motion Overview

Accelerometer

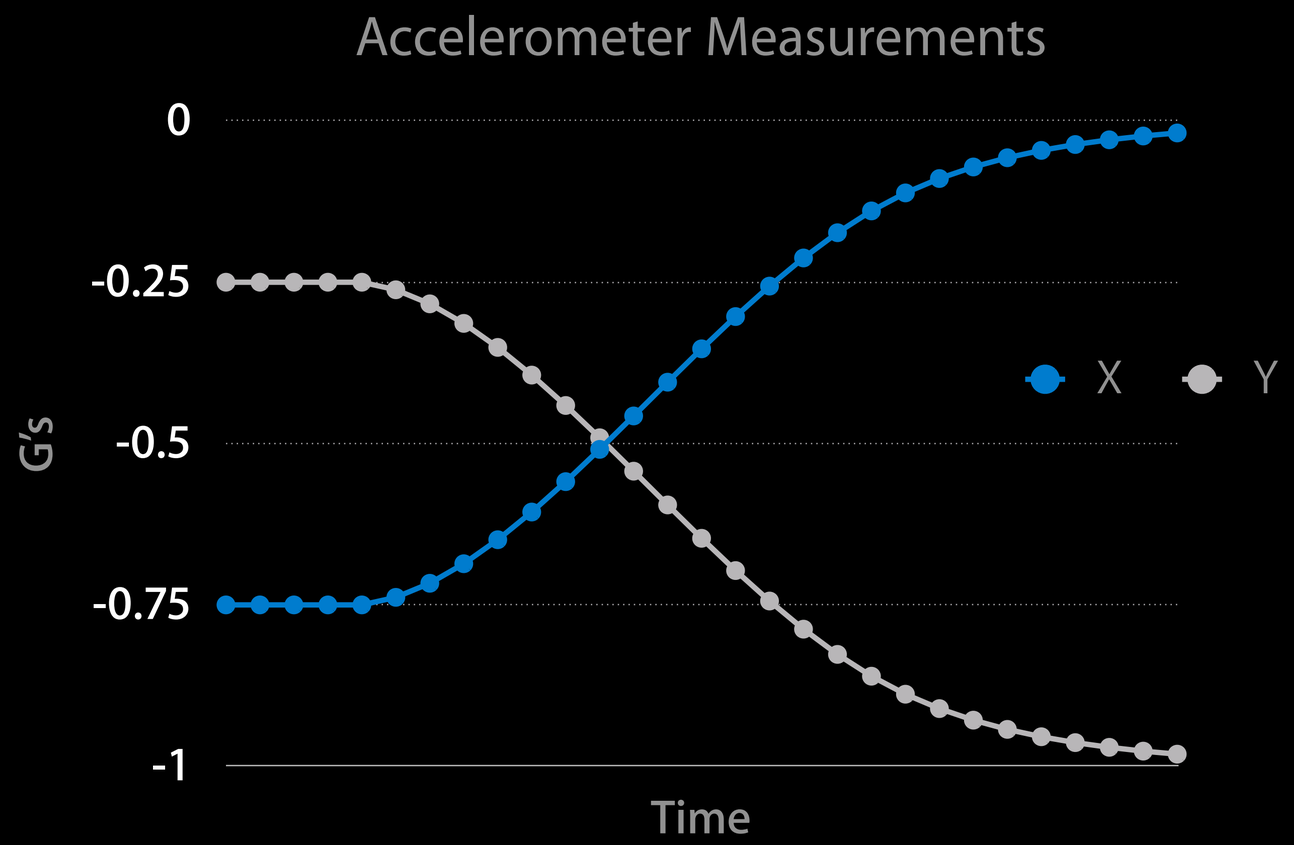
Measures gravity + accelerations

- Good for
  - Tip and tilt
  - Horizontal acceleration
- Strengths
  - Power
  - Responsiveness



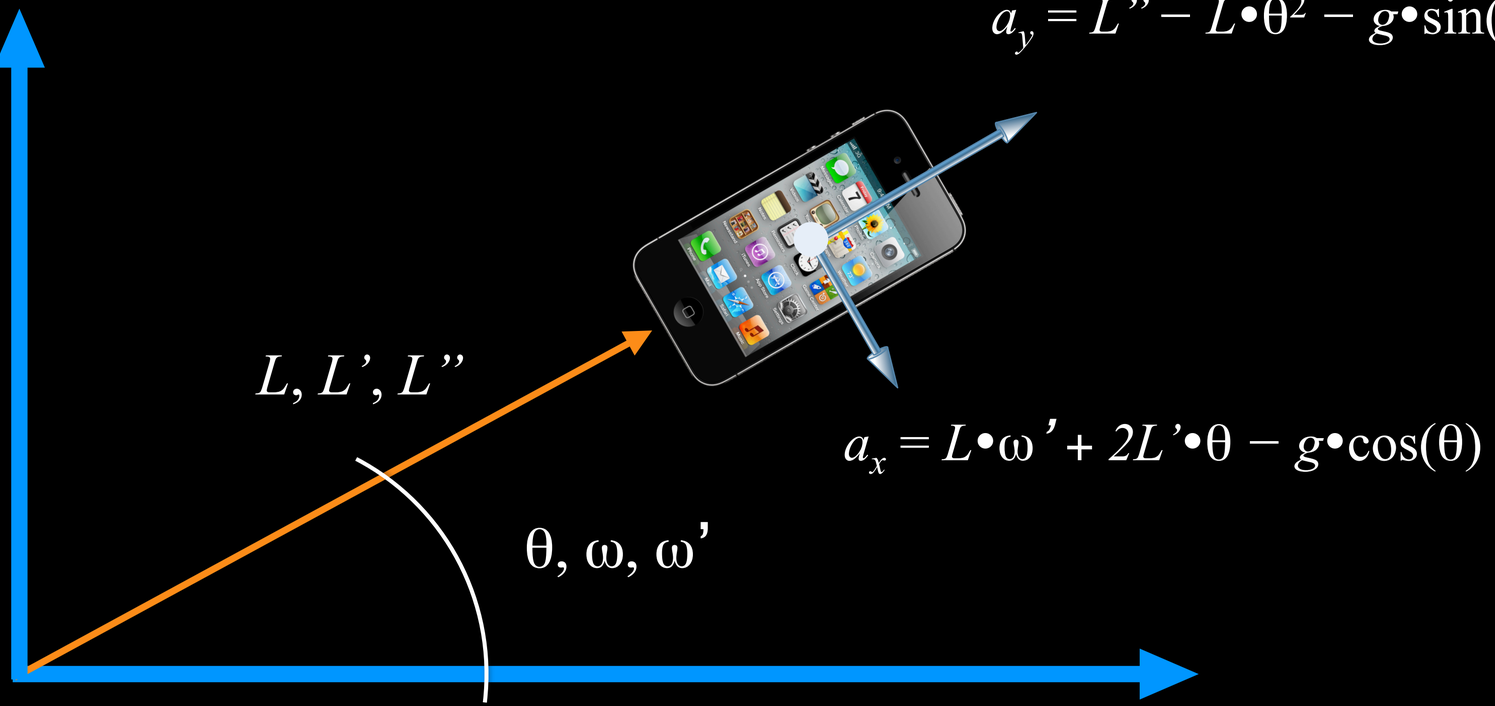
Ambiguity

2D trajectory on X-Y plane



Translation and Rotation

Accelerations have multiple components

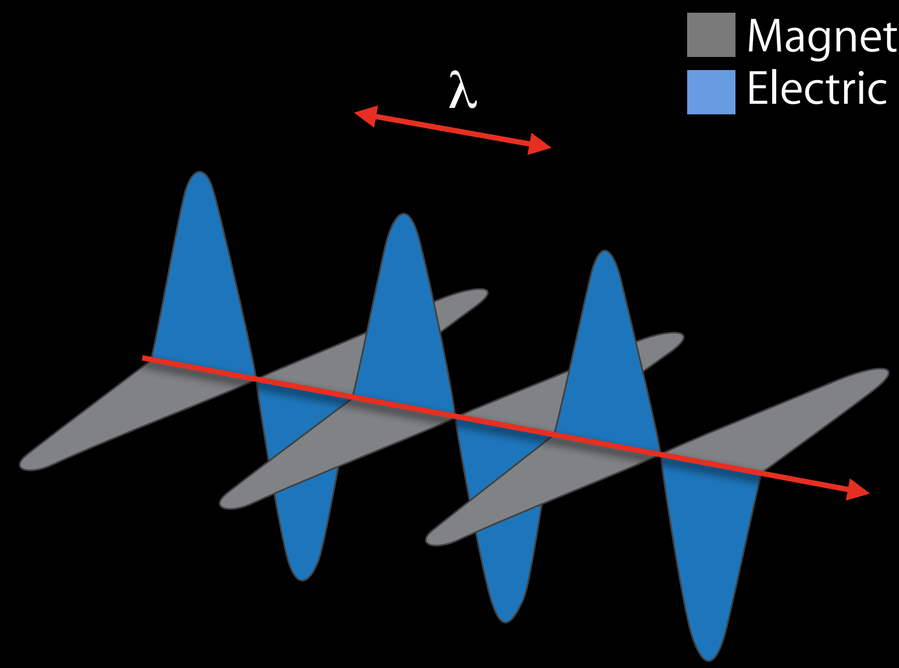


Magnetometer

Measures magnetic field

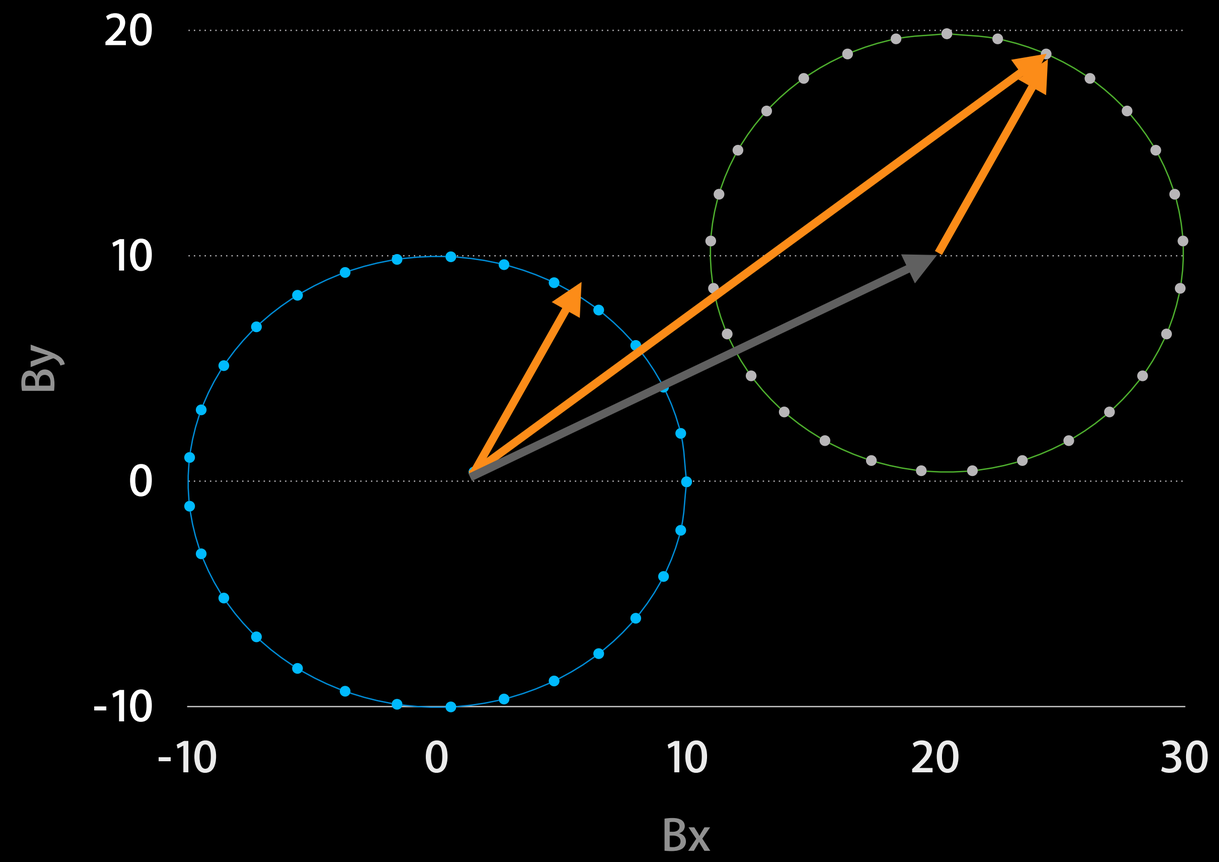
- Good for
  - Heading
  - AC field

Electromagnetic Wave



Interference

Local magnetic fields



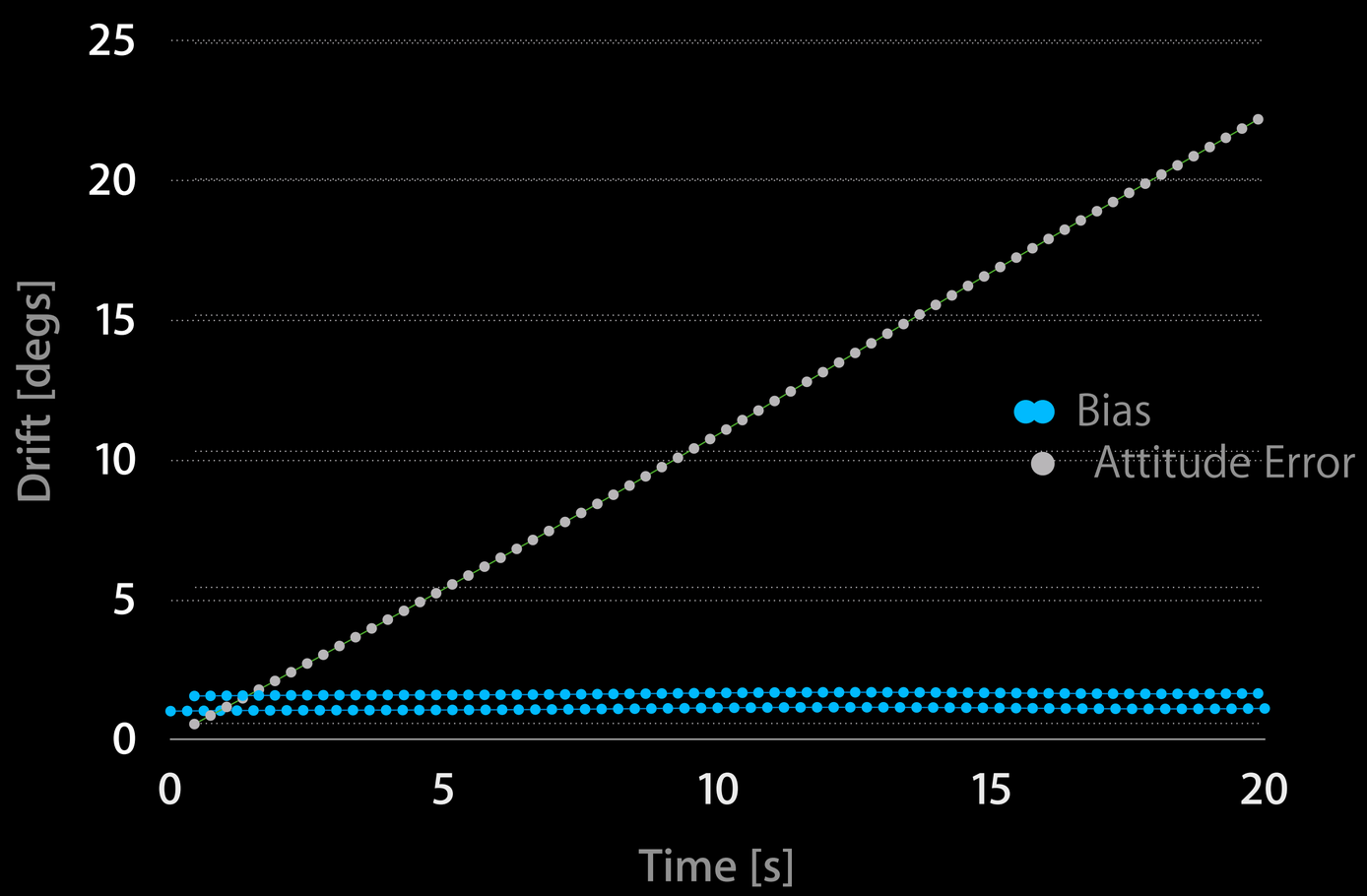
Gyro

Measures rotation rate

- Good for
  - Free-space motion
- Strengths
  - Responsiveness
  - Dynamic range
  - Precision

Bias

Attitude error grows over time



Sensor Fusion

Pointing stability + responsiveness

+

+

Key Takeaways

Select the right tool

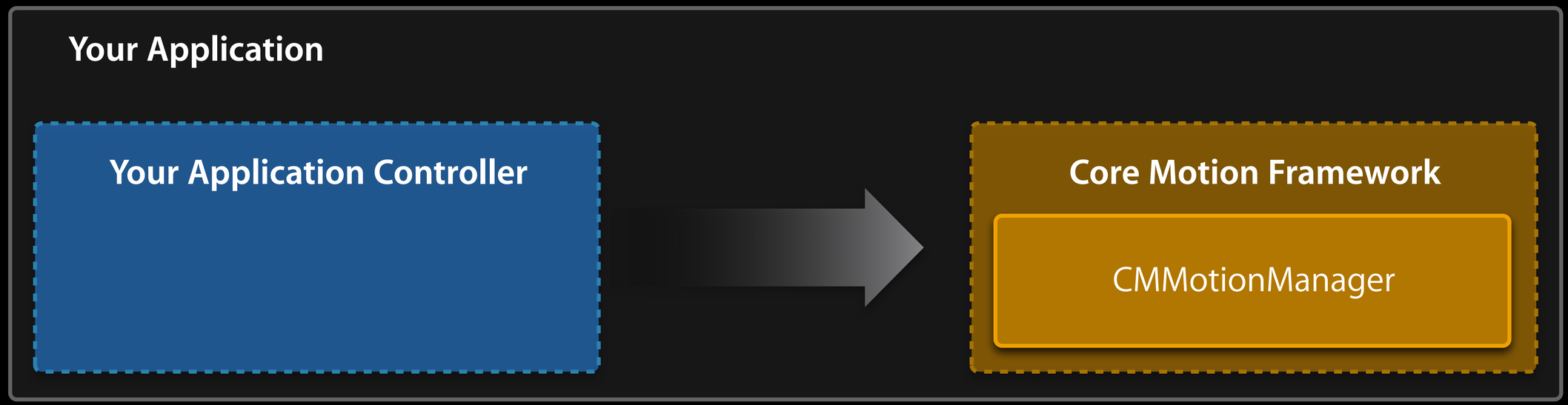
	Attitude	Heading	Spin	Shake
Accelerometer	⚠			⚠
Magnetometer		⚠		
Gyroscope			✓	
Sensor Fusion	✓	✓	✓	✓

Deep Dive

Closer look at Core Motion

Core Motion Objects

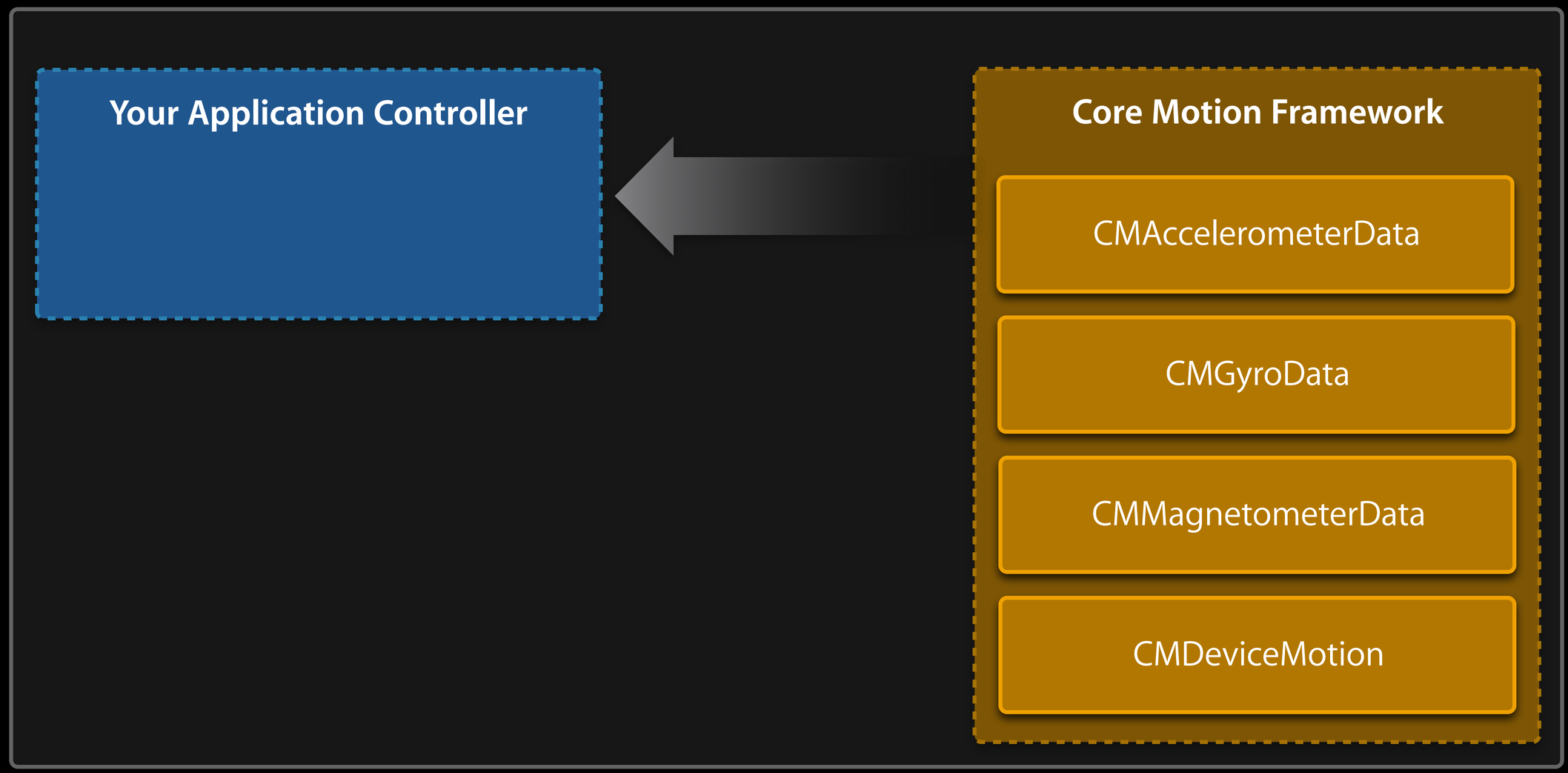
Motion manager



Set Update Intervals
Start/Stop Updates

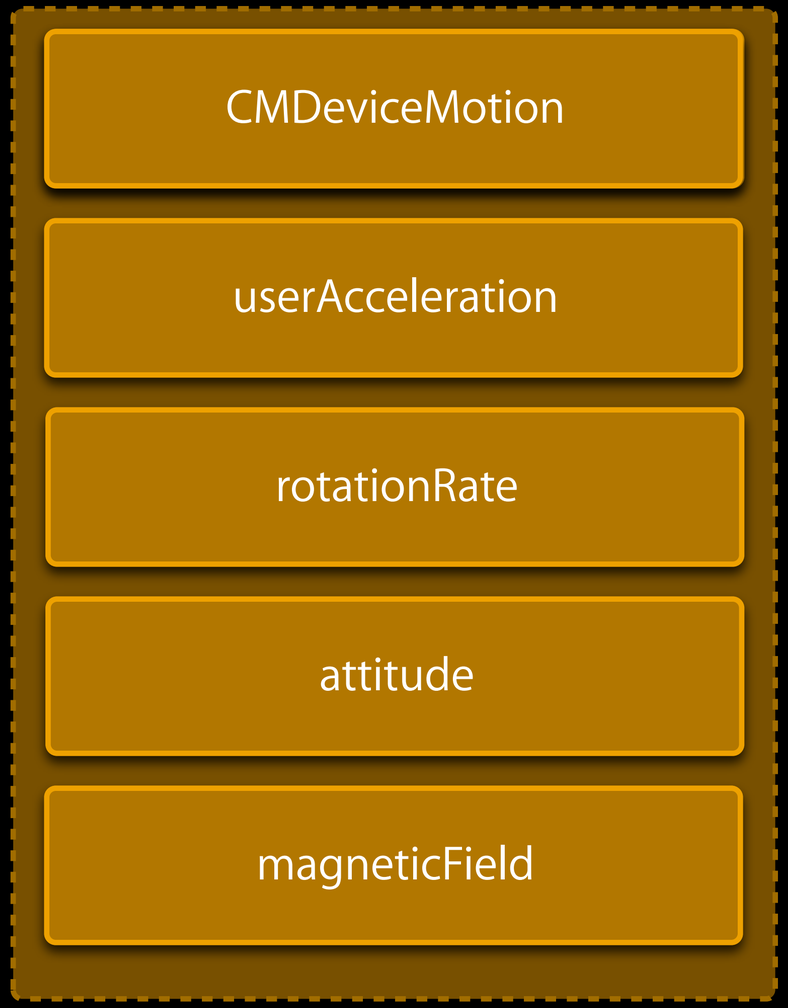
Core Motion Objects

Data objects



Sensor Fusion

CMDeviceMotion



User Acceleration

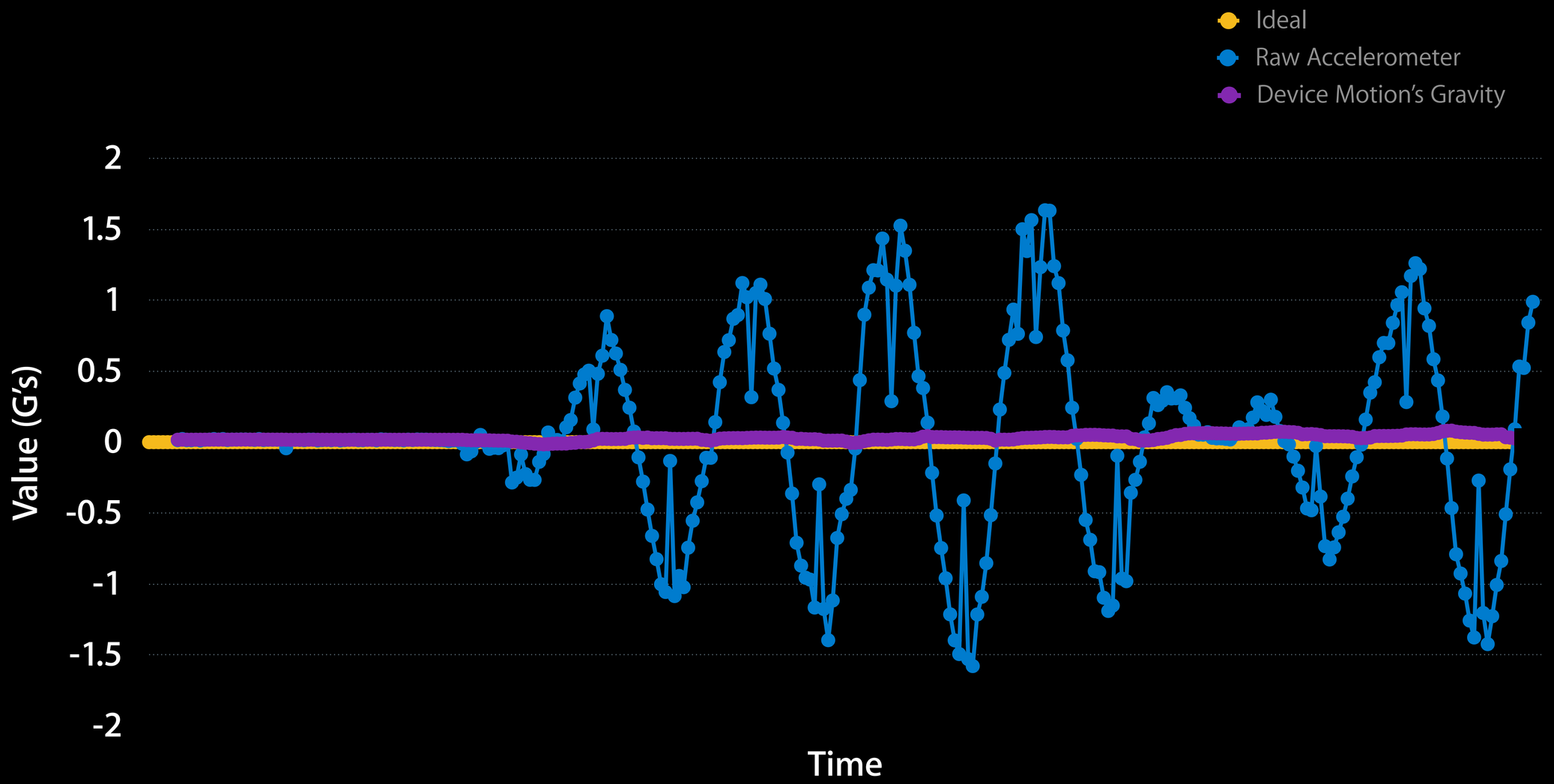
`CMAcceleration userAcceleration`

- Inertial acceleration seen by device
  - Sans gravity

```
// Units are in G's
typedef struct {
  double x;
  double y;
  double z;
} CMAcceleration;
```

Gravity

CMAcceleration gravity



Rotation Rate

`CMRotationRate rotationRate`

- Device's spin
  - Corrected for bias

```
// Units are in rads/sec
typedef struct {
  double x;
  double y;
  double z;
} CMRotationRate;
```

Magnetic Field

CMCalibratedMagneticField magneticField

- Nearby magnetic field
  - Corrected for bias
  - Filtered

```
typedef struct {
  CMMagneticField field;
  CMMagneticFieldCalibrationAccuracy accuracy;
} CMCalibratedMagneticField;
```

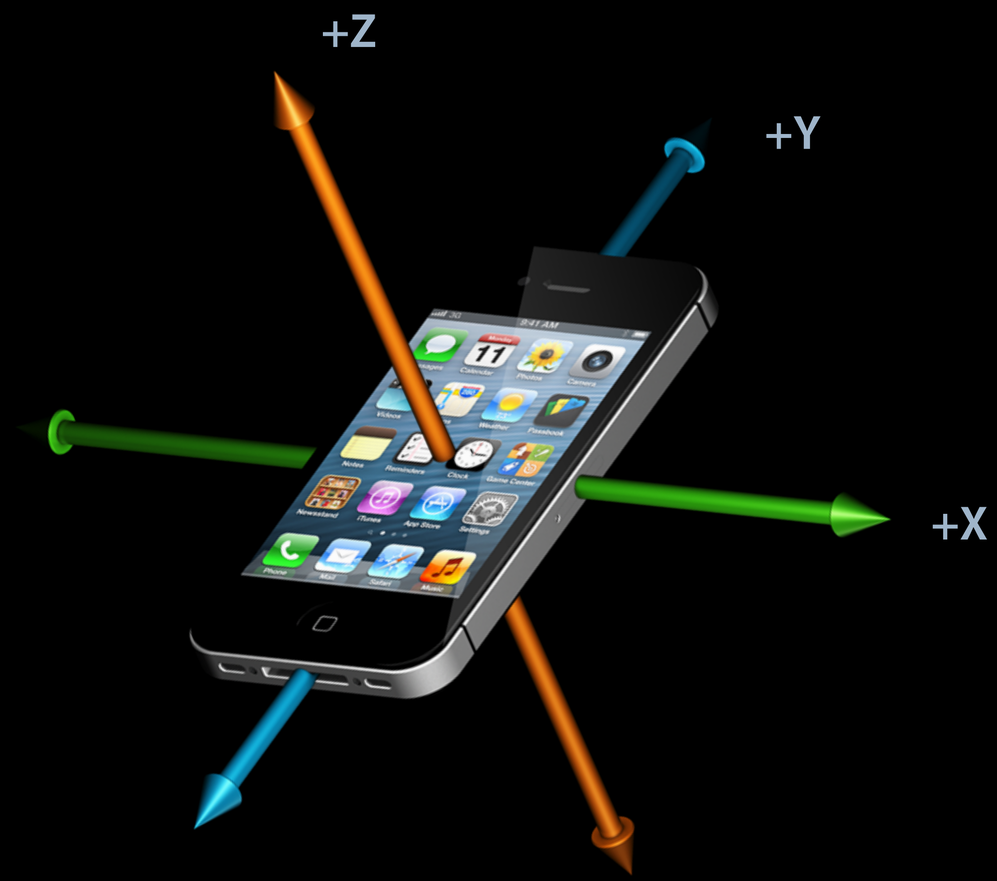
Magnetic Field

```
// Units are in uT
typedef struct {
  double x;
  double x;
  double z;
} CMMagneticField;

typedef enum {
  CMMagneticFieldCalibrationAccuracyUncalibrated = -1,
  CMMagneticFieldCalibrationAccuracyLow,
  CMMagneticFieldCalibrationAccuracyMedium,
  CMMagneticFieldCalibrationAccuracyHigh,
} CMMagneticFieldCalibrationAccuracy;
```

Attitude

`CMAttitude * attitude`

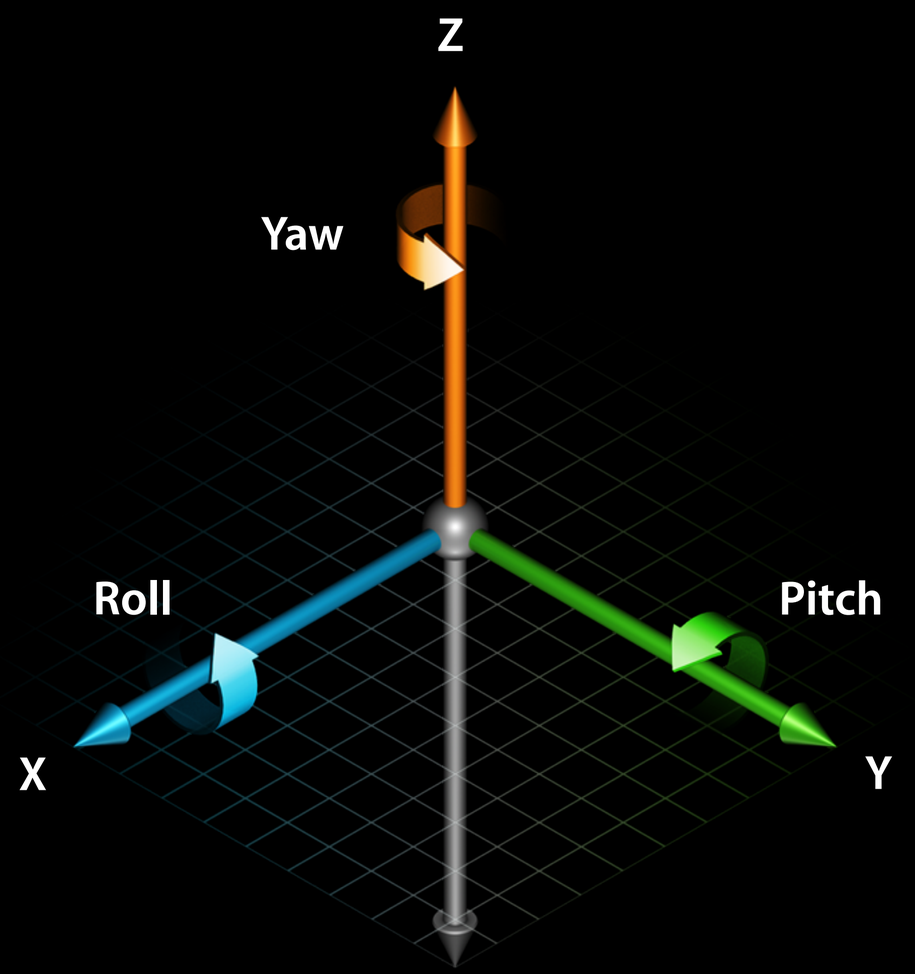
- 3D orientation
  - Euler angles
  - Quaternion
  - Rotation matrix



Euler Angles

Double roll, pitch, yaw

- Intuitive
  - Easily visualized
  - Familiar to user



Quaternions

CMQuaternion quaternion

- Elegant
  - Computational efficiency
  - Avoids gimbal lock
  - Smooth interpolation

```
typedef struct {
  double x, y, z, w;
} CMQuaternion;
```

Quaternion Utilities

Converting from quaternions

```
#include “MotionUtils.h”

- (void)rotateView
{
    // Get the current pose
    CMQuaternion qPose = motionManager.deviceMotion.quaternion;

    // Do some quaternion math...
    CMQuaternion qRotated = ...
    ...

    // Convert results into Euler Angles
    double roll = CMUtilsRollFromQuaternion(qRotated);
    double pitch = CMUtilsPitchFromQuaternion(qRotated);
    double yaw = CMUtilsYawFromQuaternion(qRotated);
}
```

Rotation Matrix

CMRotationMatrix rotationMatrix

- Convenient
  - 3D representation
  - Math, straight-up

```
typedef struct {
  double m11, m12, m13;
  double m21, m22, m23;
  double m31, m32, m33;
} CMRotationMatrix;
```

Rotation Matrix

Example

```
- (void)updateView
{
    GLKBaseEffect *effect = ...

    CMRotationMatrix r = motionManager.deviceMotion.rotationMatrix;
    GLKMatrix4 baseModelViewMatrix = GLKMatrix4Maker(r.m11, r.m22, r.m31, 0f,
                                                     r.m12, r.m22, r.m32, 0f,
                                                     r.m13, r.m23, r.m33, 0f,
                                                     0f,    0f,    0f,    1f);

    GLKMatrix4 modelViewMatrix = GLKMatrix4Identity;
    modelViewMatrix = GLKMatrix4Rotate(modelView, PI/3, 1f, 1f, 1f);

    modelViewMatrix = GLKMatrix4Multiply(baseModelViewMatrix, modelViewMatrix);

    effect.transform.modelviewMatrix = modelViewMatrix;
}
```

Reference Frame Choices

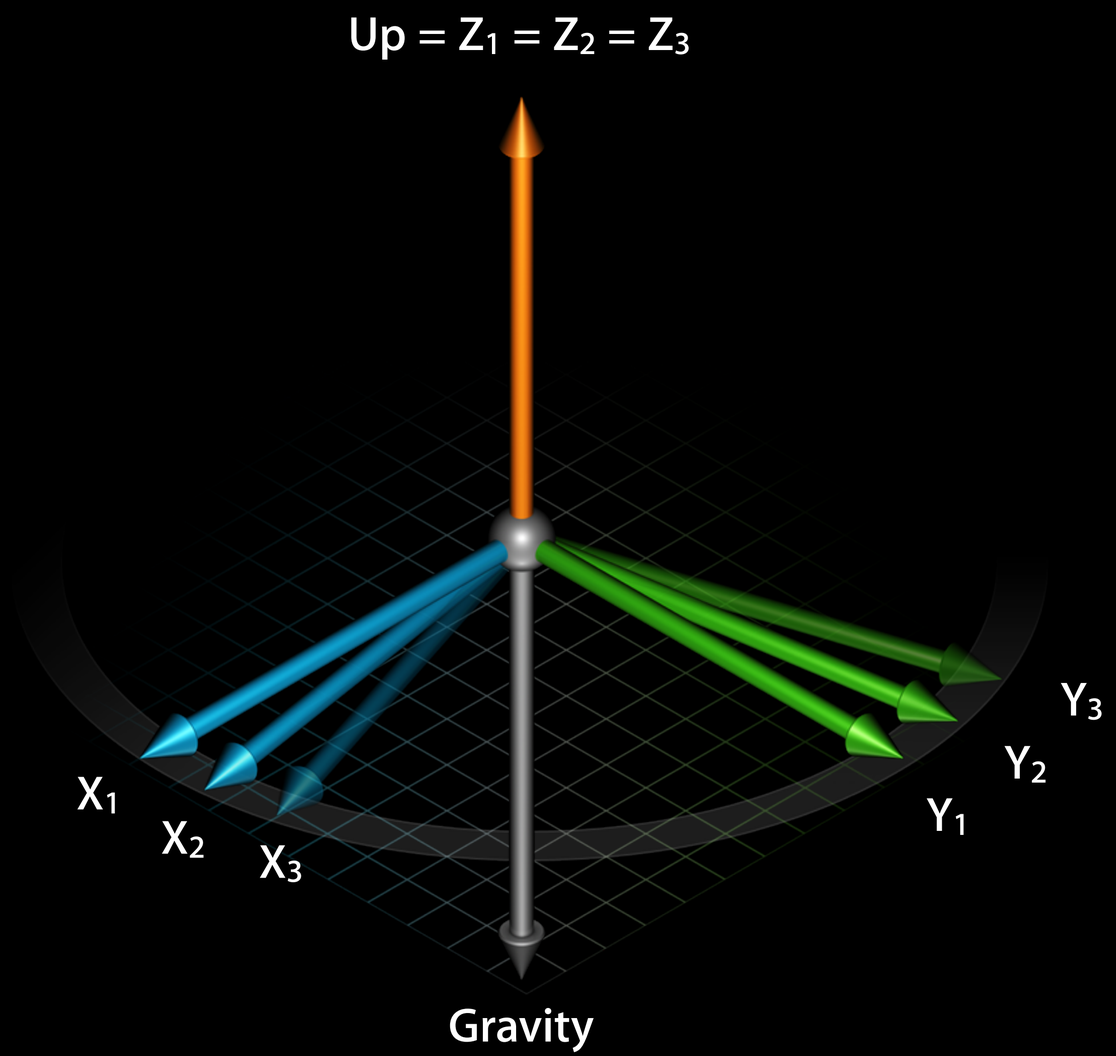
```
typedef enum {
  CMAttitudeReferenceFrameXArbitraryZVertical = 1 << 0,
  CMAttitudeReferenceFrameXArbitraryCorrectedZVertical = 1 << 1,
  CMAttitudeReferenceFrameXMagneticNorthZVertical = 1 << 2,
  CMAttitudeReferenceFrameXTrueNorthZVertical = 1 << 3
} CMAttitudeReferenceFrame;
```

```
+ (NSUInteger)availableAttitudeReferenceFrames
```

```
- (void)startDeviceMotionUpdatesUsingReferenceFrame:
```

Reference Frame Choices

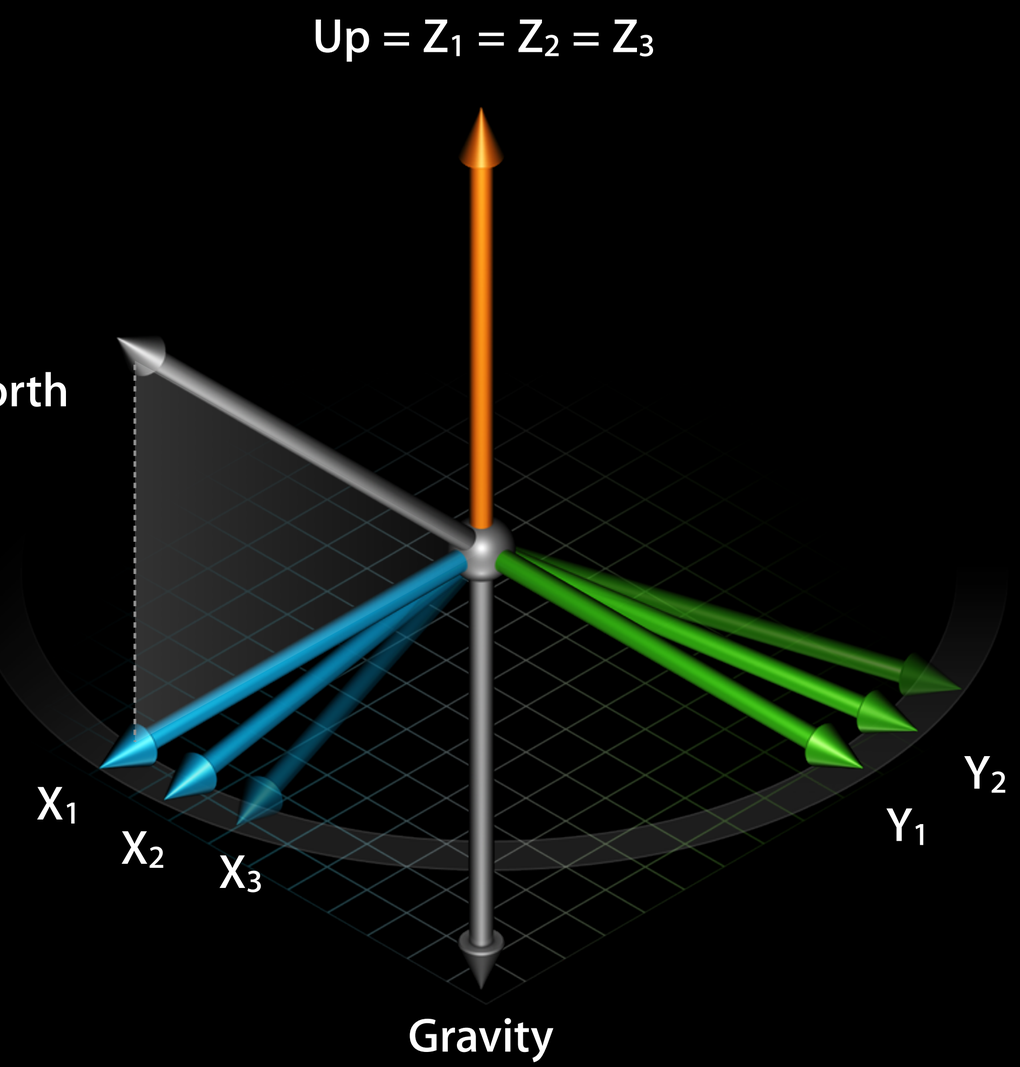
XArbitrary



Reference Frame Choices

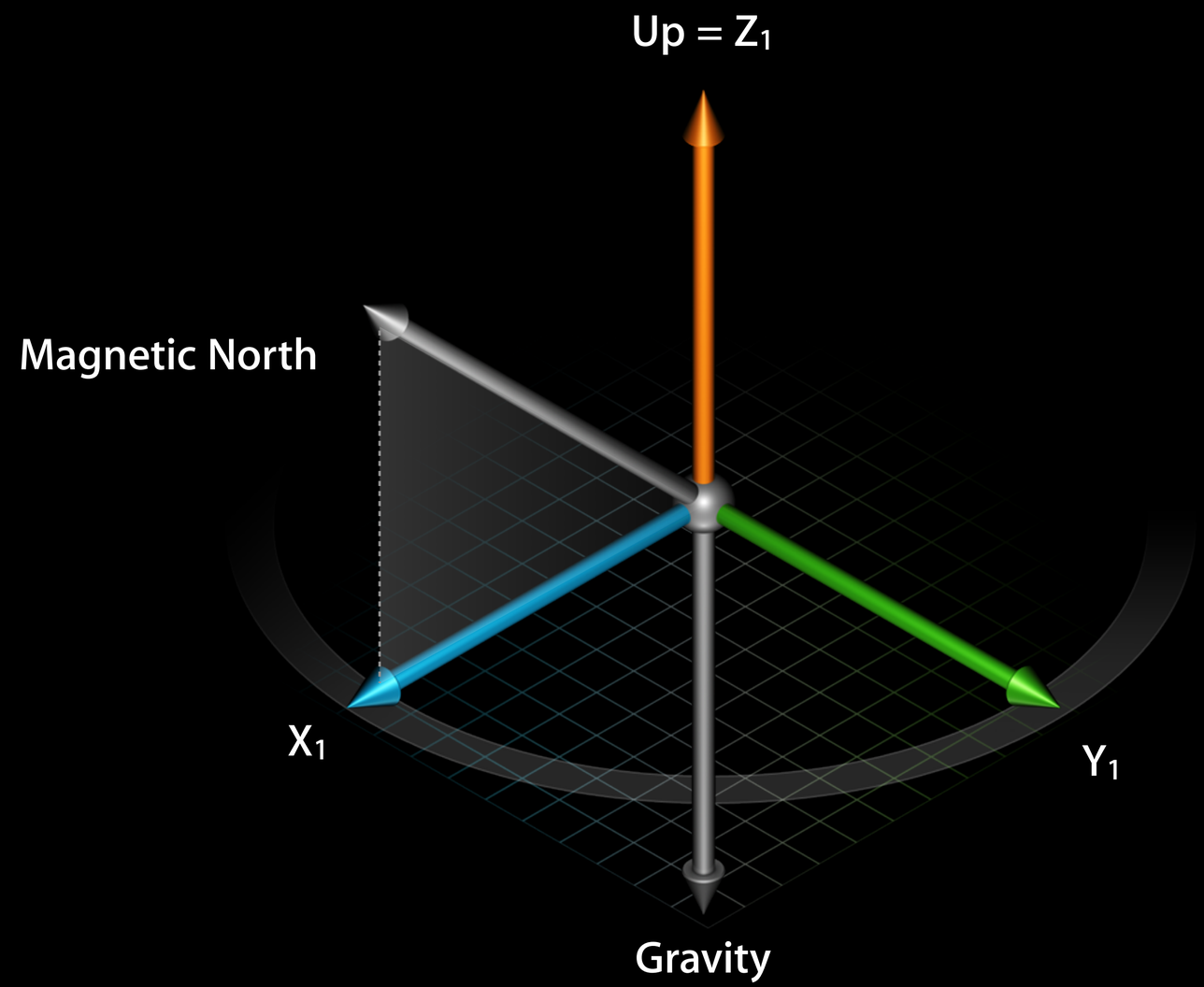
XArbitraryCorrected

- Opportunistic heading correction
  - Better longterm yaw accuracy
  - Higher CPU usage



Reference Frame Choices

XMagneticNorth



Calibration Requirements

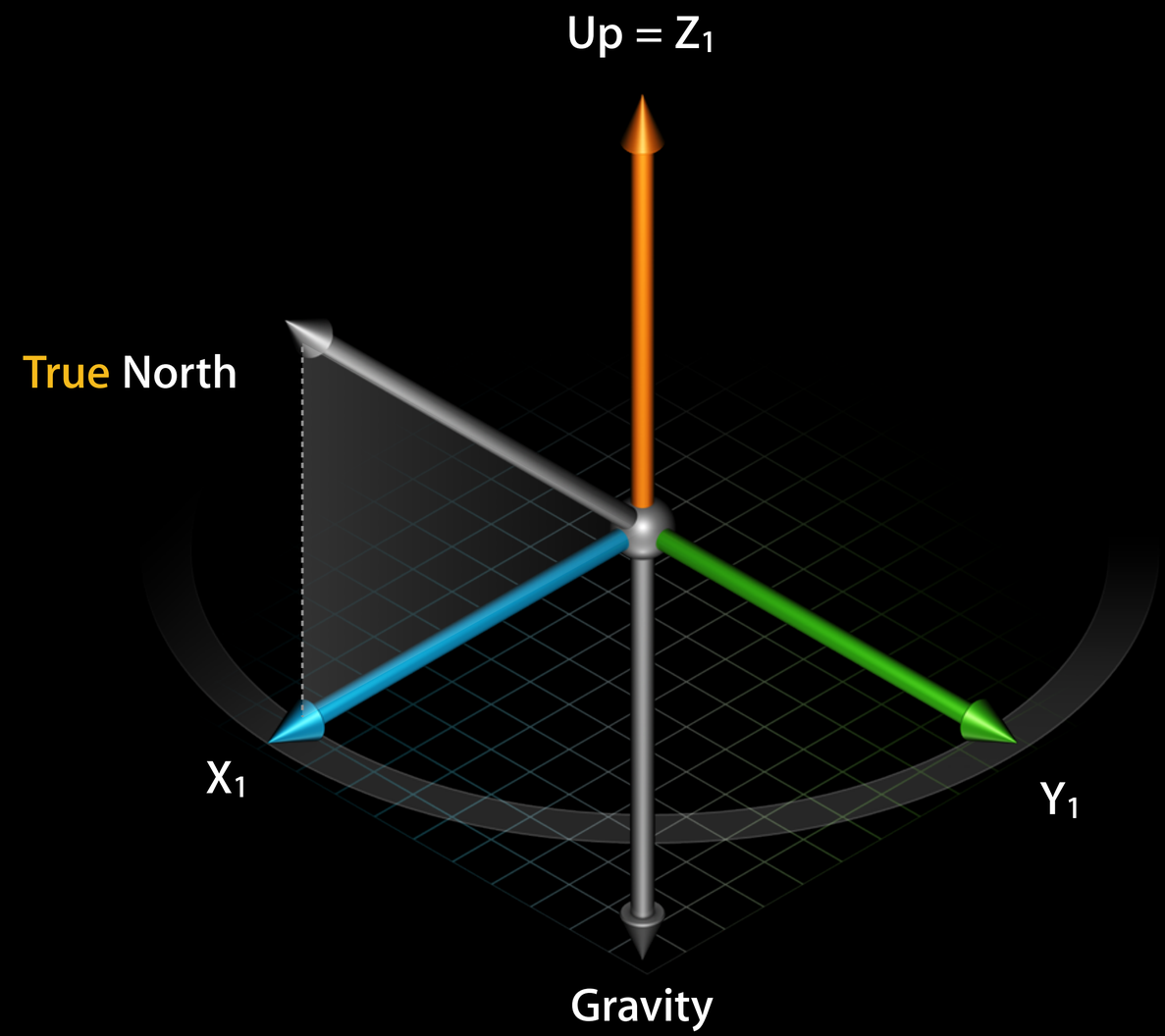
XMagneticNorth

- Magnetometer calibration may be required

```
@property(assign, nonatomic) BOOL showsDeviceMovementDisplay
```

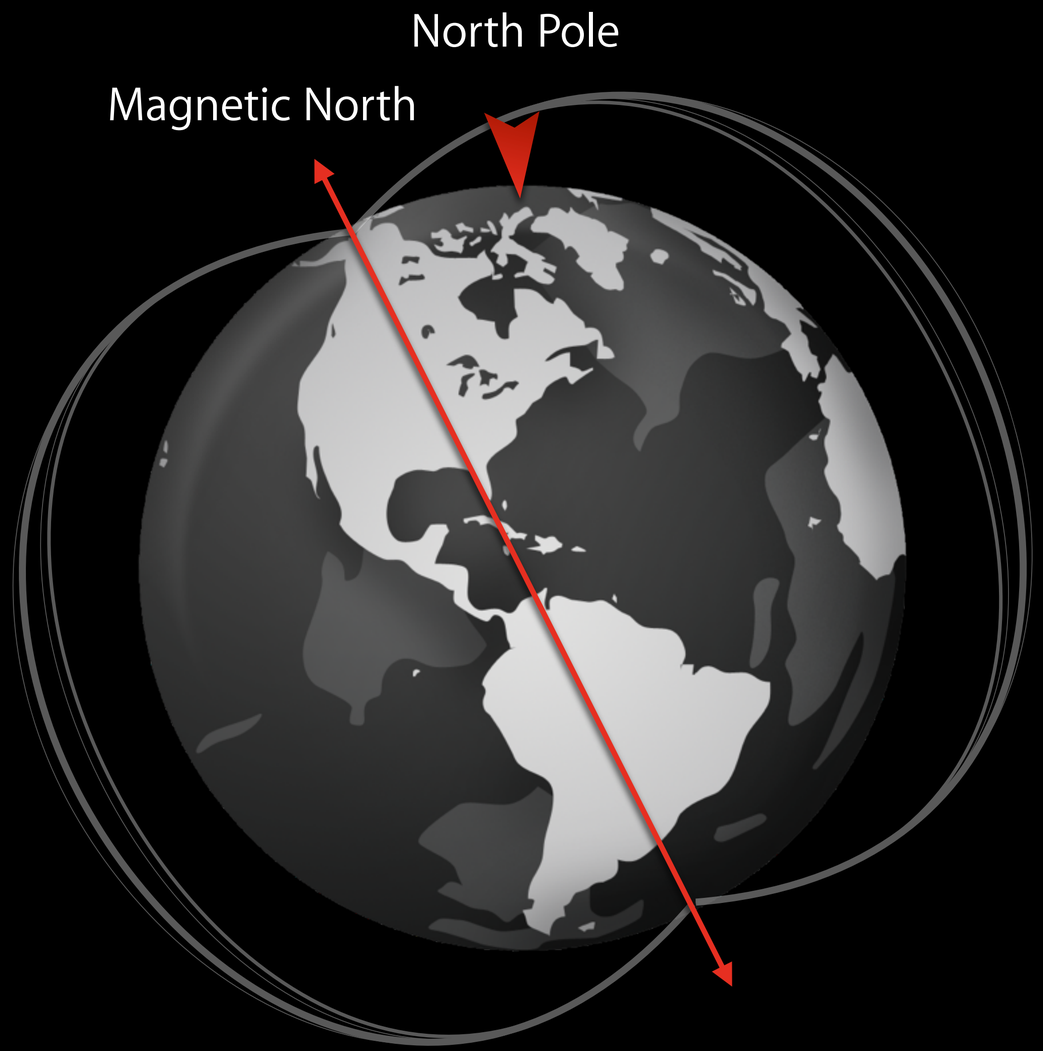
Reference Frame Choices

CMAttitudeReferenceFrameXTrueNorthZVertical



Magnetic North

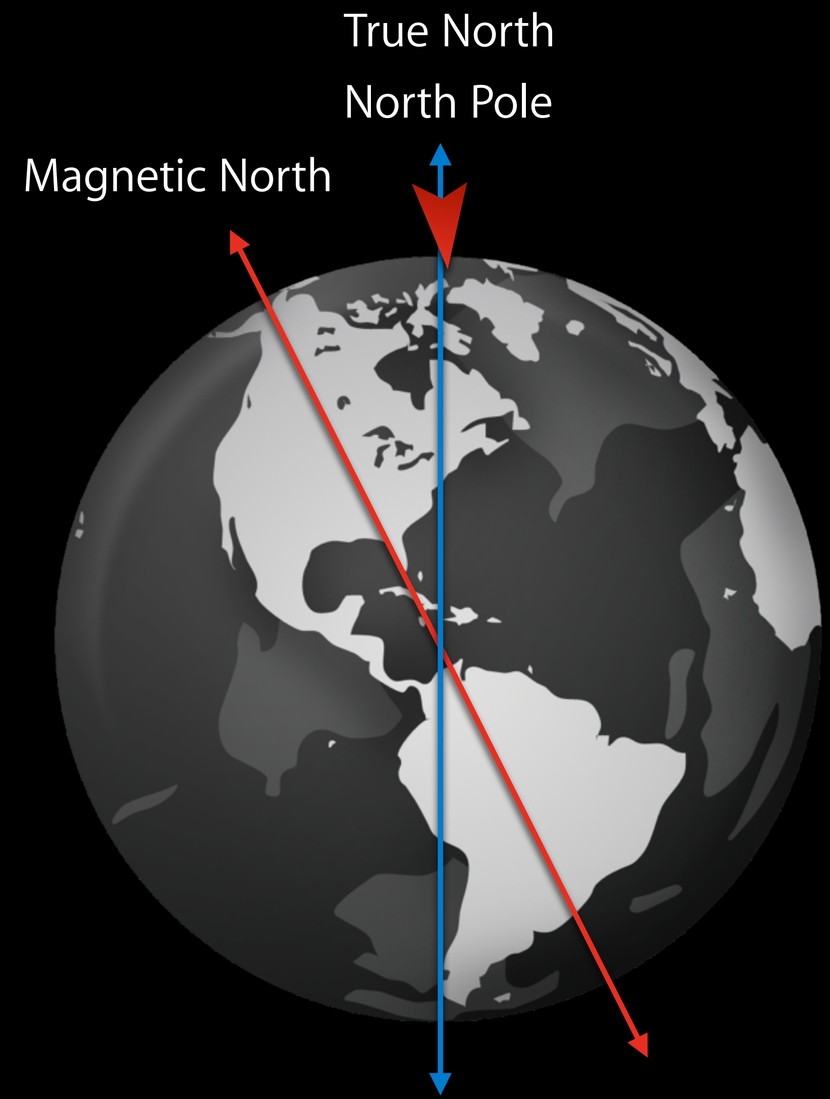
Earth's magnetic field



True North

Earth's geographic poles

- Direction we are most familiar with
  - Reference point used when creating maps
- Can calculate true north given
  - Magnetic north
  - Model of difference between true north and magnetic north around globe
  - Approximate location



Key Takeways

Sensor fusion

- Sensor fusion does the hard work
  - Separates out gravity
  - Removes gyro bias
  - Calibrates magnetometer

- Use Reference Frame to specify flavor of sensor fusion
  - XArbitrary | XArbitraryCorrected
  - XMagneticNorth | XTrueNorth

Using Core Motion

Outline For Using Core Motion

1. Setup
   - Select raw sensor or sensor fusion flavor you want
   - Define update interval
2. Retrieve data
   - Pull data from Core Motion
   - Have Core Motion push data to you
3. Cleanup

Step 1. Setup

```
- (void)startAnimation
{
    // Create a CMMotionManager instance
    motionManager = [[CMMotionManager alloc] init];

    // Ensure that the data we're interested in is available
    if (!motionManager.isAccelerometerAvailable) {
        // Fail gracefully
    }

    // Set the desired update interval (60Hz in this case)
    motionManager.accelerometerUpdateInterval = 1.0 / 60.0;

    // Start updates
    // Note: We could call the following here instead:
    // [motionManager startAccelerometerUpdatesToQueue:withHandler:]
    [motionManager startAccelerometerUpdates];
}
```

Step 2. Retrieve Data

```
- (void)drawView:(id)sender
{
    CMAccelerometerData *newestAccel = motionManager.accelerometerData;
    // ...
}
```

Retrieving Data

Framework

- Core Motion creates its own thread
  - Handle raw data from sensors
  - Run sensor fusion algorithms

Retrieving Data

Your application

- Push
  - You provide NSOperationQueue and block
  - Only your block will execute on your threads
- Pull
  - Periodically ask CMMotionManager for latest sample
  - Core Motion will never interrupt your threads

Retrieving Data

Push vs. Pull

	Advantages	Disadvantages	Recommendation
Push	Never miss a sample	Increased overhead Often best to drop samples	Data collection apps
Pull	More efficient Less code required	May need additional timer	Most apps and games

Step 3. Cleanup

```
- (void)stopAnimation
{
    [motionManager stopAccelerometerUpdates];
    motionManager = nil;

    //...
}
```

Best Practices

Getting the most out of Core Motion

1. Change reference frames
2. Characterize sensors
3. Optimize update interval
4. Common pitfalls

Select Suitable Reference

multiplyByInverseOfAttitude

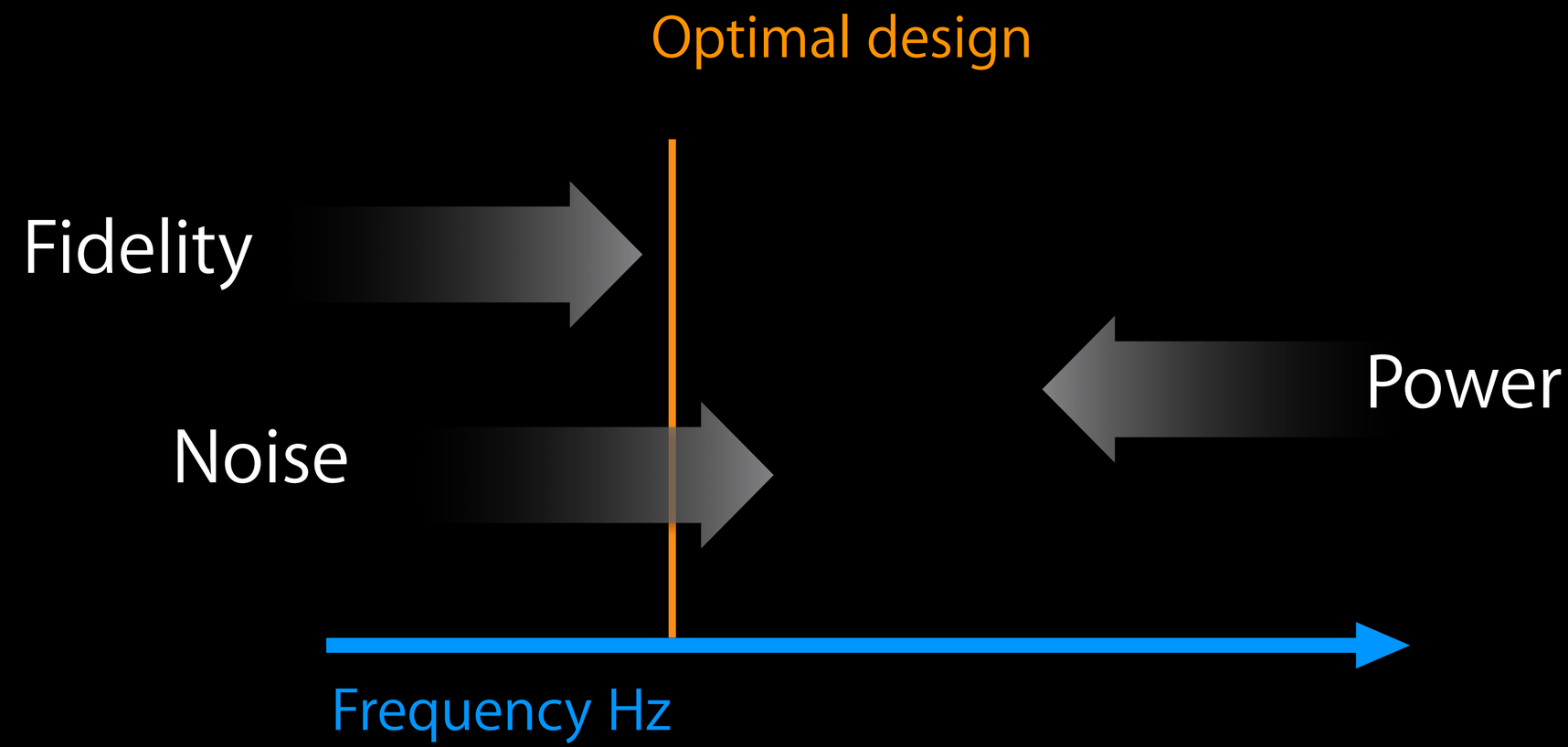
```
- (void)onResetReference
{
    fReferenceAttitude = motionManager.deviceMotion.attitude;
}
```

```
- (void)drawView:(id)sender
{
    ...
    attitude = motionManager.deviceMotion.attitude;
    [attitude multiplyByInverseOfAttitude: fReferenceAttitude];
    ...
}
```

Characterize Sensors

- Ensure that performance meets your requirements
  - Dynamic range
  - Sensitivity
  - Dynamic response
  - Noise
  - Bias

Optimize Update Interval



Use Correct Timestamp

```
- (void)drawView:(id)sender
{
    CMAccelerometerData *newestAccel = motionManager.accelerometerData;

    // Do this
    CFAbsoluteTime time = newestAccel.timestamp;

    // Instead of this
    // CFAbsoluteTime time = CFAbsoluteTimeGetCurrent();
}
```

Do Not Assume Update Interval

```
static const int accelerometerUpdateInterval = 1.0 / 50.0;


// Push semantics
- (void)onAccelerometerData(CMAccelerometerData *accelerometerSample)
{
    ...
    CFAbsoluteTime timeStamp = accelerometerSample.timestamp;

    // Compute a moving average to smooth over jitter
    fDeltaTime = Average(timeStamp - lastTimeStamp);

    ...
}
```

Wait for Data

```
- (void)drawView:(id)sender
{
    CMAccelerometerData *newestAccel = motionManager.accelerometerData;

    if (newestAccel != nil)  {
        // Do something with the measurement
    }
}
```

Avoid Integration

```
- (void)drawView:(id)sender
{
    CMAccelerometerData *newestAccel = motionManager.accelerometerData;

    // Danger! This will be unstable:
    fVelocity.x = fVelocity.x + newestAccel.acceleration.x * fDeltaTime
    fVelocity.y = fVelocity.y + newestAccel.acceleration.y * fDeltaTime
    fVelocity.z = fVelocity.z + newestAccel.acceleration.z * fDeltaTime
}
```

Check for Existence

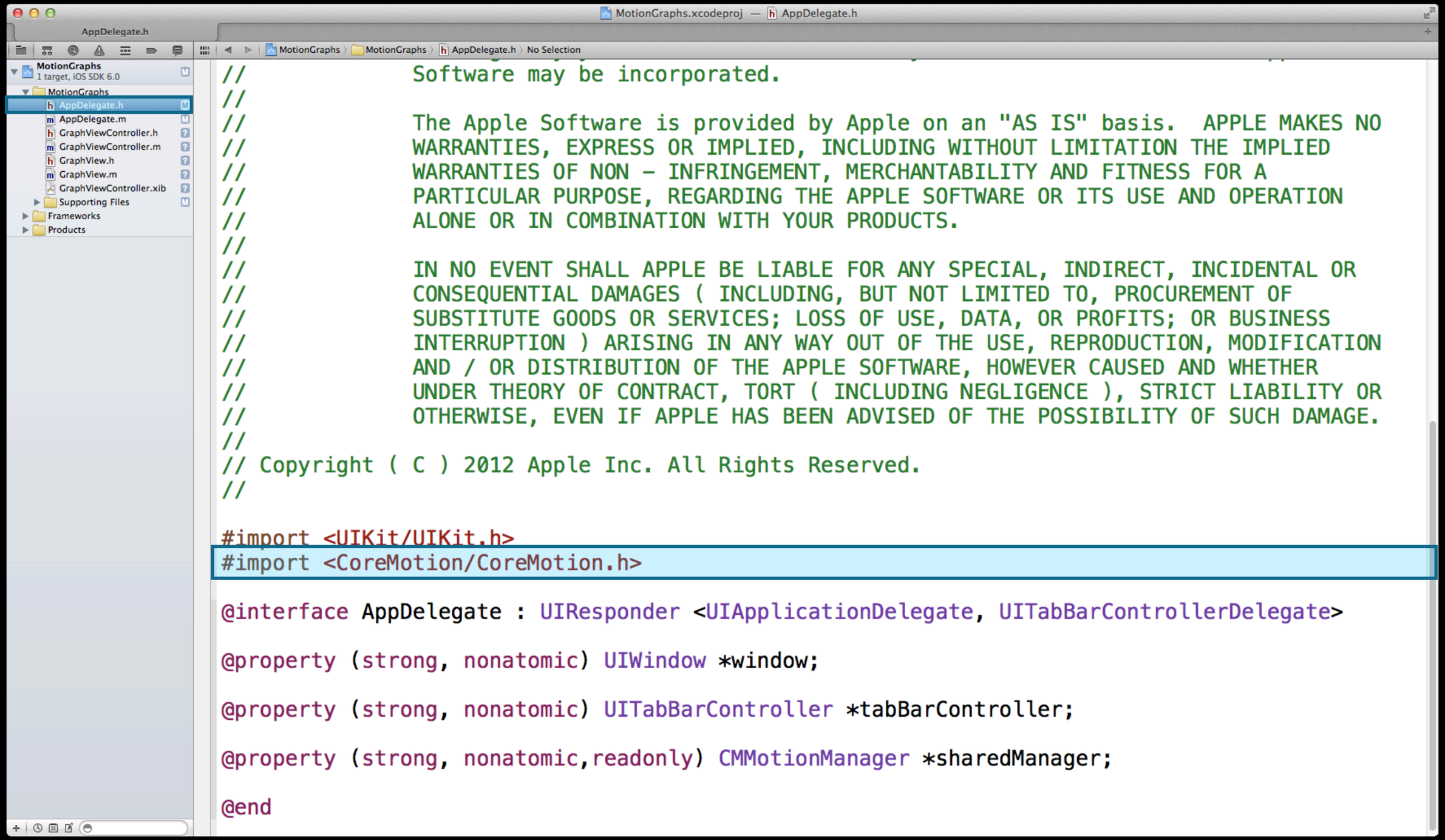
```
- (void)startAnimation
{
    ...
    // For example, if using raw magnetometer
    if (!motionManager.isMagnetometerAvailable) {
        // Fail gracefully
    }

    // For example, if requesting true-north referenced sensor fusion
    if (!([CMMotionManager availableAttitudeReferenceFrames] &
          CMAttitudeReferenceFrameXTrueNorthZVertical)) {
        // Fail gracefully
    }

    ...
}
```

Coding Examples

MotionGraphs

Visualize Core Motion output

- Concepts
  - Use push semantics
  - Manipulate update interval


MotionGraphs.xcodeproj — AppDelegate.h
AppDelegate.h
MotionGraphs › MotionGraphs › AppDelegate.h › No Selection
MotionGraphs
1 target, iOS SDK 6.0
MotionGraphs
AppDelegate.h
AppDelegate.m
GraphViewController.h
GraphViewController.m
GraphView.h
GraphView.m
GraphViewController.xib
Supporting Files
Frameworks
Products
// Software may be incorporated.
//
// The Apple Software is provided by Apple on an "AS IS" basis. APPLE MAKES NO
// WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED
// WARRANTIES OF NON - INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A
// PARTICULAR PURPOSE, REGARDING THE APPLE SOFTWARE OR ITS USE AND OPERATION
// ALONE OR IN COMBINATION WITH YOUR PRODUCTS.
//
// IN NO EVENT SHALL APPLE BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL OR
// CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
// SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
// INTERRUPTION) ARISING IN ANY WAY OUT OF THE USE, REPRODUCTION, MODIFICATION
// AND / OR DISTRIBUTION OF THE APPLE SOFTWARE, HOWEVER CAUSED AND WHETHER
// UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR
// OTHERWISE, EVEN IF APPLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Copyright (C) 2012 Apple Inc. All Rights Reserved.
//
#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>
@interface AppDelegate : UIResponder <UIApplicationDelegate, UITabBarControllerDelegate>
@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) UITabBarController *tabBarController;
@property (strong, nonatomic,readonly) CMMotionManager *sharedManager;
@end

MotionGraphs.xcodeproj — AppDelegate.m

AppDelegate.m

MotionGraphs › MotionGraphs › AppDelegate.m › No Selection

MotionGraphs
1 target, iOS SDK 6.0

- MotionGraphs
  - AppDelegate.h
  - AppDelegate.m
  - GraphViewController.h
  - GraphViewController.m
  - GraphView.h
  - GraphView.m
  - GraphViewController.xib
  - Supporting Files
- Frameworks
- Products

```
@interface AppDelegate()
{
    CMMotionManager *motionmanager;
}
@end

@implementation AppDelegate

- (CMMotionManager *)sharedManager
{
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        motionmanager = [[CMMotionManager alloc] init];
    });
    return motionmanager;
}
```

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *
    )launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    UIViewController *viewControllerAccelerometer, *viewControllerGyro, *
        viewControllerDeviceMotion;
    viewControllerAccelerometer = [[GraphViewController alloc] initWithMotionDataType:
        kMotionDataTypeAccelerometerData];
    viewControllerAccelerometer.title = @"Accelerometer";
    viewControllerGyro = [[GraphViewController alloc] initWithMotionDataType:
        kMotionDataTypeGyroData];
    viewControllerGyro.title = @"Gyro";
    viewControllerDeviceMotion = [[GraphViewController alloc] initWithMotionDataType:
        kMotionDataTypeDeviceMotion];
```

MotionGraphs.xcodeproj — GraphViewController.m

GraphViewController.m

MotionGraphs › MotionGraphs › GraphViewController.m › No Selection

- MotionGraphs
  1 target, iOS SDK 6.0
  - MotionGraphs
    - AppDelegate.h
    - AppDelegate.m
    - GraphViewController.h
    - GraphViewController.m
    - GraphView.h
    - GraphView.m
    - GraphViewController.xib
    - Supporting Files
  - Frameworks
  - Products

```
    primaryGraphLabel.text = [graphTitles objectAtIndex:sender.selectedSegmentIndex];
}

- (IBAction)onSliderValueChanged:(UISlider *)sender
{
    [self startUpdatesWithMotionDataType:graphDataSource andSliderValue:(int)(sender.value *
        100)];
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [self startUpdatesWithMotionDataType:graphDataSource andSliderValue:(int)
        (updateIntervalSlider.value * 100)];
}

-(void)viewDidLoad
{
    [super viewDidLoad];

    mManager = [(AppDelegate *)[[UIApplication sharedApplication] delegate] sharedManager];

    updateIntervalSlider.value = 0.0f;

    if (graphDataSource != kMotionDataTypeDeviceMotion) {
        segmentedControl.hidden = YES;
    } else {
        graphTitles = @[@"motion.attitude", @"motion.rotationRate", @"motion.gravity",
            @"motion.userAcceleration"];

        GraphView *attitudeGraph = primaryGraph;
        GraphView *rotationRateGraph = [[GraphView alloc] initWithFrame:primaryGraph.frame];
```

MotionGraphs.xcodeproj — GraphViewController.m

GraphViewController.m

MotionGraphs › MotionGraphs › GraphViewController.m › No Selection

MotionGraphs
1 target, iOS SDK 6.0

- MotionGraphs
  - AppDelegate.h
  - AppDelegate.m
  - GraphViewController.h
  - GraphViewController.m
  - GraphView.h
  - GraphView.m
  - GraphViewController.xib
  - Supporting Files
- Frameworks
- Products

```
- (void)startUpdatesWithMotionDataType:(MotionDataType)type andSliderValue:(int)sliderValue
{
    NSTimeInterval updateInterval;
    NSTimeInterval delta = 0.005;
    switch (graphDataSource) {
        case kMotionDataTypeAccelerometerData:
        {
            updateInterval = accelerometerMin + delta * sliderValue;

            if ([mManager isAccelerometerAvailable] == YES) {
                [mManager setAccelerometerUpdateInterval:updateInterval];
                [mManager startAccelerometerUpdatesToQueue:[NSOperationQueue mainQueue]
                    withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {
                    [primaryGraph addX:accelerometerData.acceleration.x y:accelerometerData.
                        acceleration.y z:accelerometerData.acceleration.z];
                    [self setLabelValueX:accelerometerData.acceleration.x y:accelerometerData.
                        acceleration.y z:accelerometerData.acceleration.z];
                }];
            }
            primaryGraphLabel.text = @"accelerometerData.acceleration";
            break;
        }
        case kMotionDataTypeGyroData:
        {
            updateInterval = gyroMin + delta * sliderValue;

            if ([mManager isGyroAvailable] == YES) {
                [mManager setGyroUpdateInterval:updateInterval];
                [mManager startGyroUpdatesToQueue:[NSOperationQueue mainQueue] withHandler:^
                    (CMGyroData *gyroData, NSError *error) {
```

MotionGraphs.xcodeproj — GraphViewController.m

GraphViewController.m

MotionGraphs › MotionGraphs › GraphViewController.m › No Selection

MotionGraphs
1 target, iOS SDK 6.0

- MotionGraphs
  - AppDelegate.h
  - AppDelegate.m
  - GraphViewController.h
  - GraphViewController.m
  - GraphView.h
  - GraphView.m
  - GraphViewController.xib
  - Supporting Files
- Frameworks
- Products

```objc
- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
    [self stopUpdatesWithMotionDataType:graphDataSource];
}

- (void)stopUpdatesWithMotionDataType:(MotionDataType)type
{
    switch (graphDataSource) {
        case kMotionDataTypeAccelerometerData:
            if ([mManager isAccelerometerActive] == YES) {
                [mManager stopAccelerometerUpdates];
            }
            break;
        case kMotionDataTypeGyroData:
            if ([mManager isGyroActive] == YES) {
                [mManager stopGyroUpdates];
            }
            break;
        case kMotionDataTypeDeviceMotion:
            if ([mManager isDeviceMotionActive] == YES) {
                [mManager stopDeviceMotionUpdates];
            }
            break;
        default:
            break;
    }
}

@end
```

Stars

Locate objects within 3D sphere

- Concepts
  - Use pull semantics
  - Chain transformations

More Information

Allan Schaffer
Graphics Evangelist
aschaffer@apple.com

Documentation
Event Handling Guide for iPhoneOS
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Optimizing 2D Graphics and Animation Performance	Mission Tuesday 3:15PM
Advances in OpenGL and OpenGL ES	Pacific Heights Wednesday 2:00PM
OpenGL ES Tools and Techniques	Pacific Heights Wednesday 3:15PM
Integrating Your Games with Game Center	Pacific Heights Wednesday 4:30PM

Labs

Core Motion Lab	Graphics, Media & Games Lab D Friday 11:30AM
Game Center Lab	Graphics, Media & Games Lab C Friday 9:00AM

WWDC2012

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.