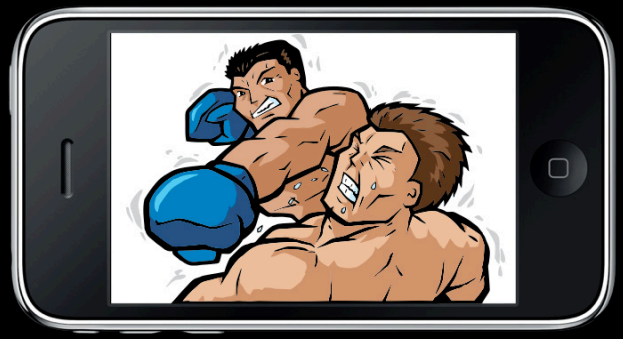
Sensing Device Motion in iOS 4

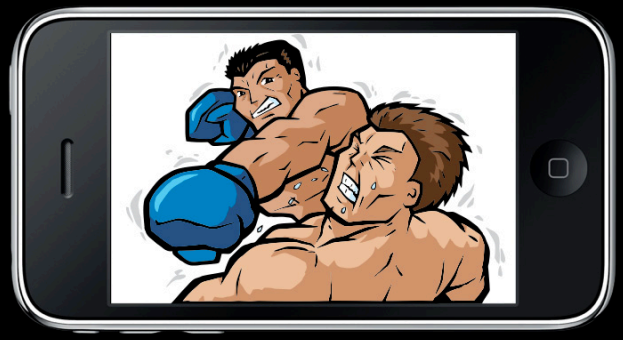
Chris Moore
iPhone OS Software Engineer

Agenda

1. What are the new features?
2. How can I access them?
3. Deep dive
4. Let's code!



Rotation = Attack



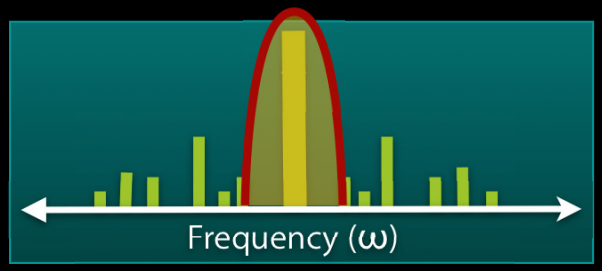
Fast translation = Dodge

Accelerometer

Measures gravity and user acceleration

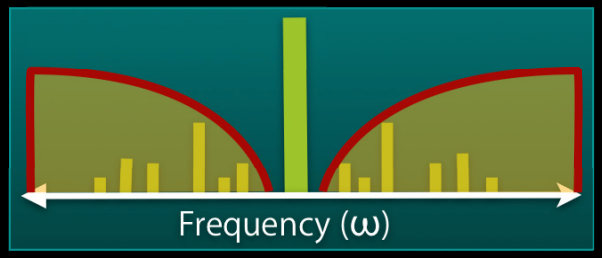
Accelerometer Responsibilities

- Gravity for rotations
- User acceleration for shakes



Low-pass filter isolates gravity

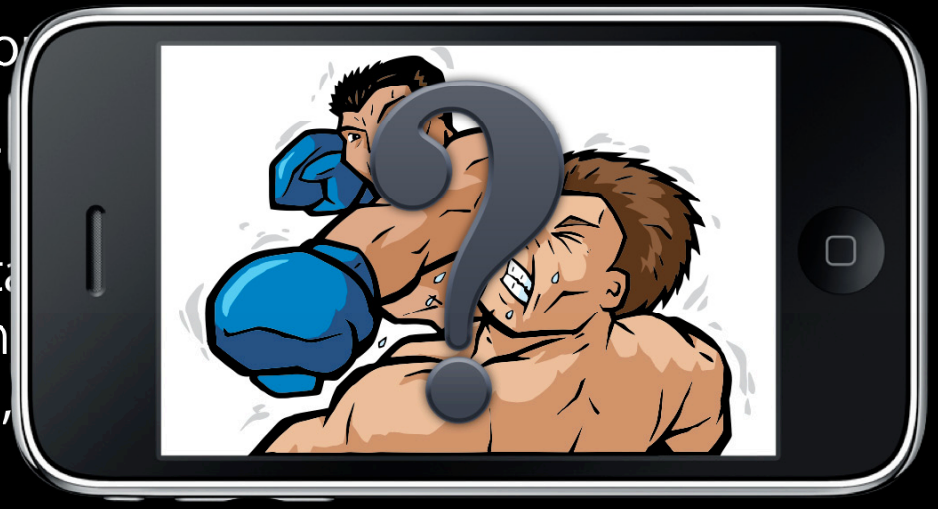
- No rotation about gravity



High-pass filter isolates
user acceleration/"shake"


3G
9:42 AM
Messages
Calendar
Tuesday
9
Photos
Camera
YouTube
Stocks
Maps
280
Weather
73°
Voice Memos
Notes
Clock
Calculator
Settings
iTunes
App Store
Compass
Phone
Mail
Safari
iPod

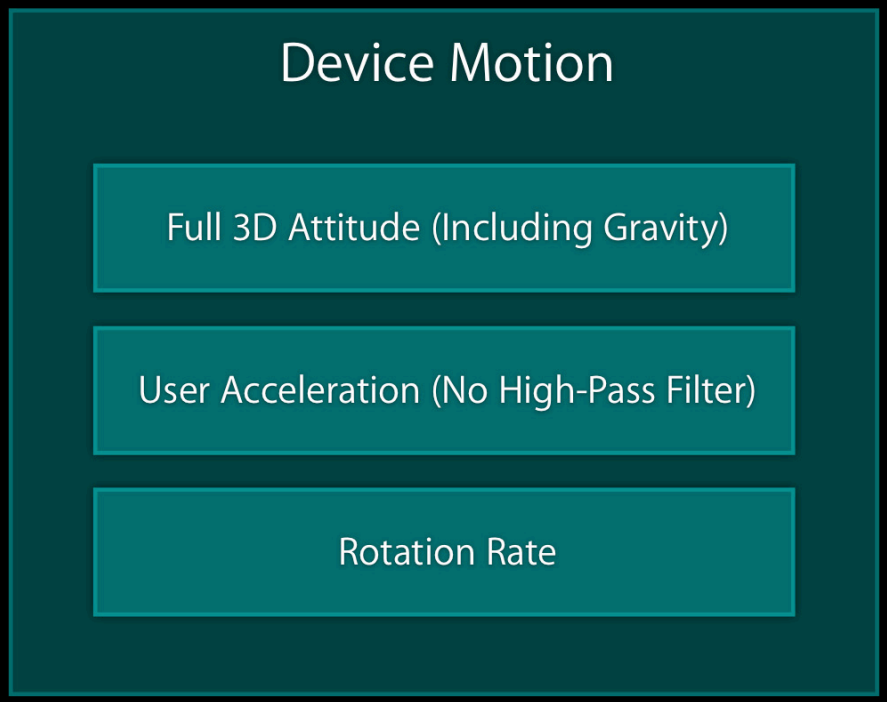
1. Rotation abo
2. Accurate for
3. Accurate rot
   the face of h
   acceleration,
   vice versa



Magnetometer

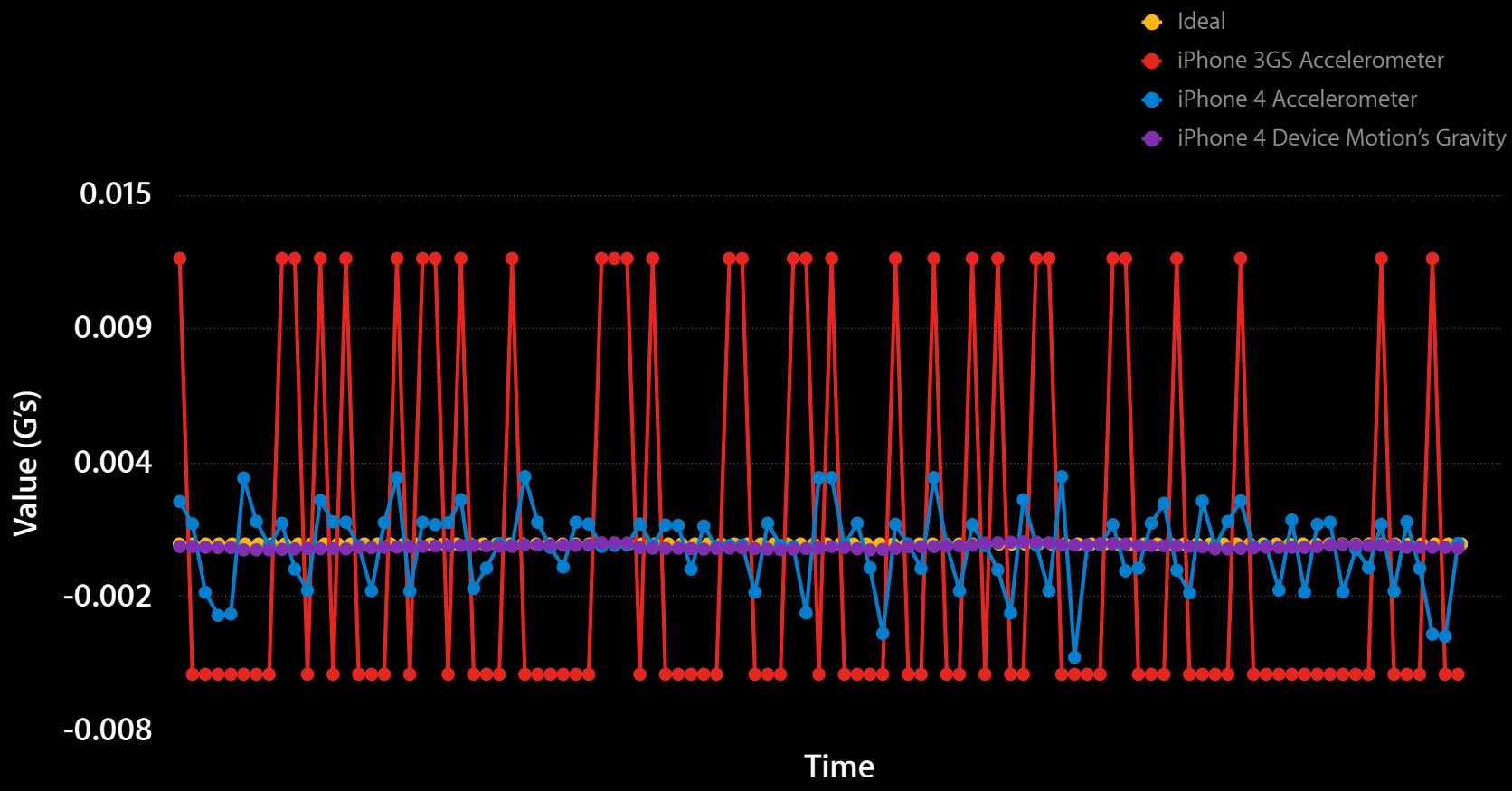
Gyroscope

Gyro and Accelerometer Fusion



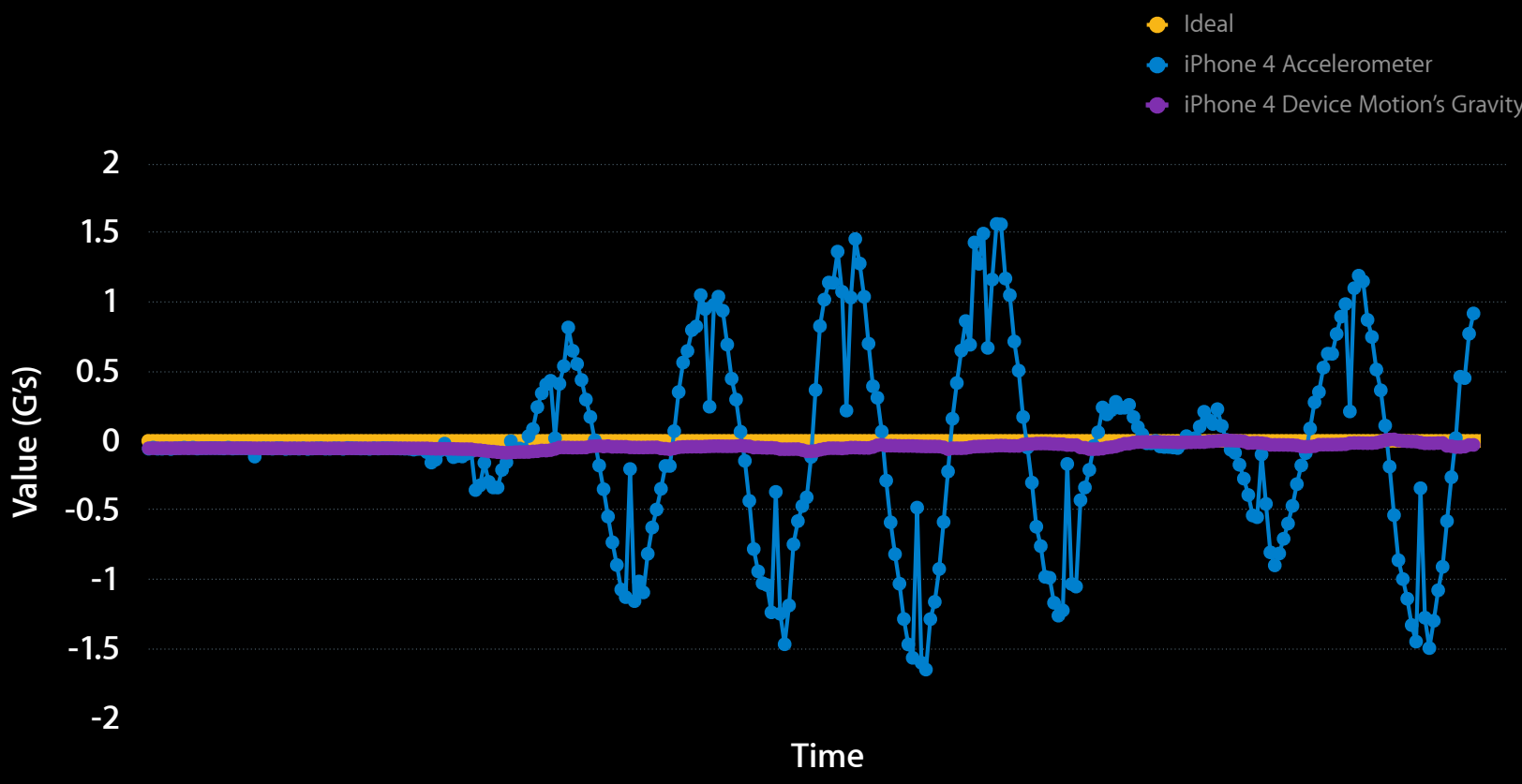
Accelerometer vs. Device Motion's Gravity

Noise

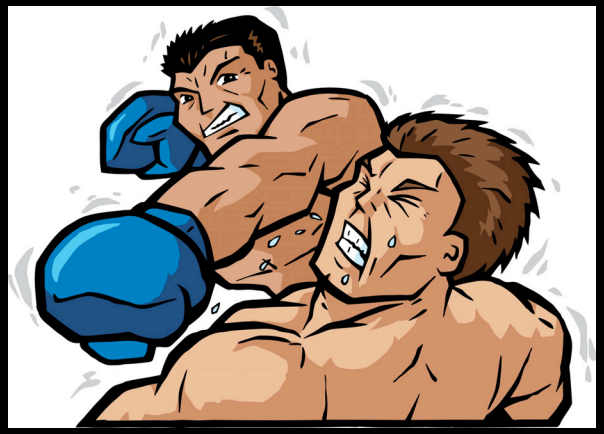


Accelerometer vs. Device Motion's Gravity

Sensitivity to user acceleration



1. Rotation abo
2. Accurate for
3. Accurate rot
   the face of h
   acceleration,
   vice versa



Demo

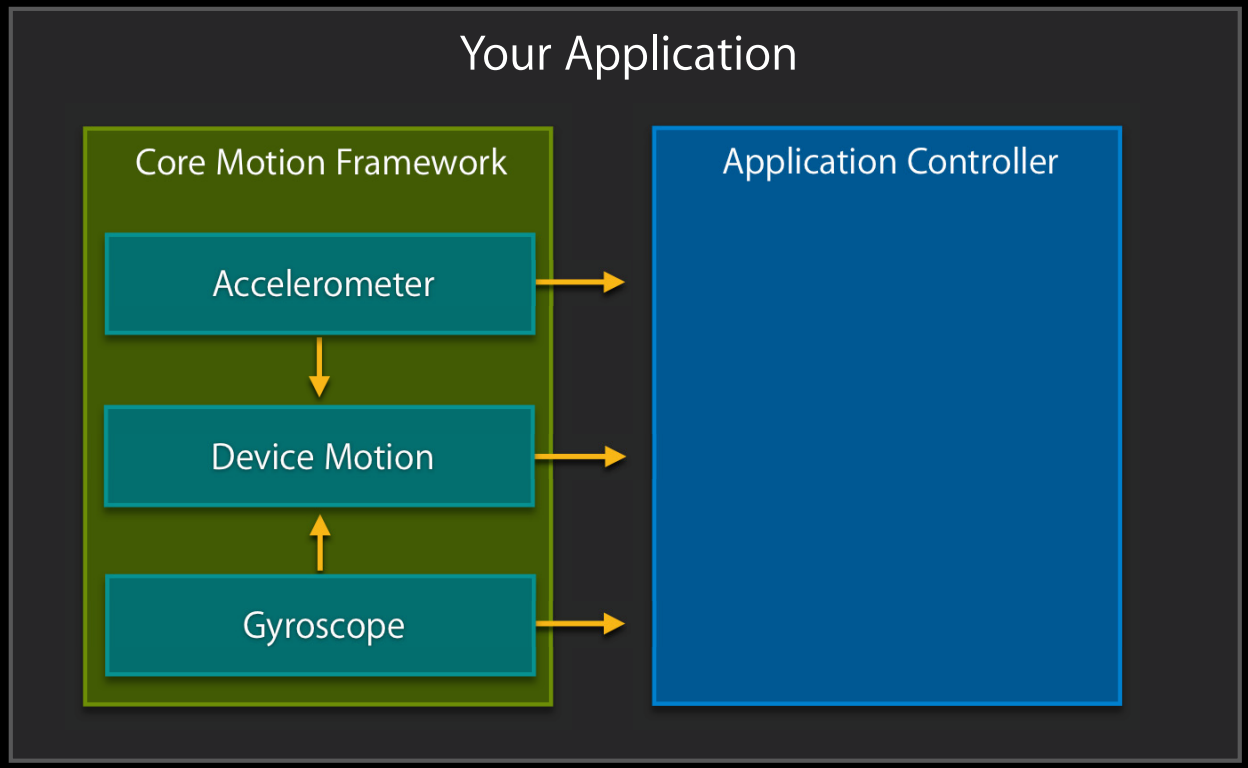
Patrick Piemonte
iPhone OS Software Engineer

Using Core Motion

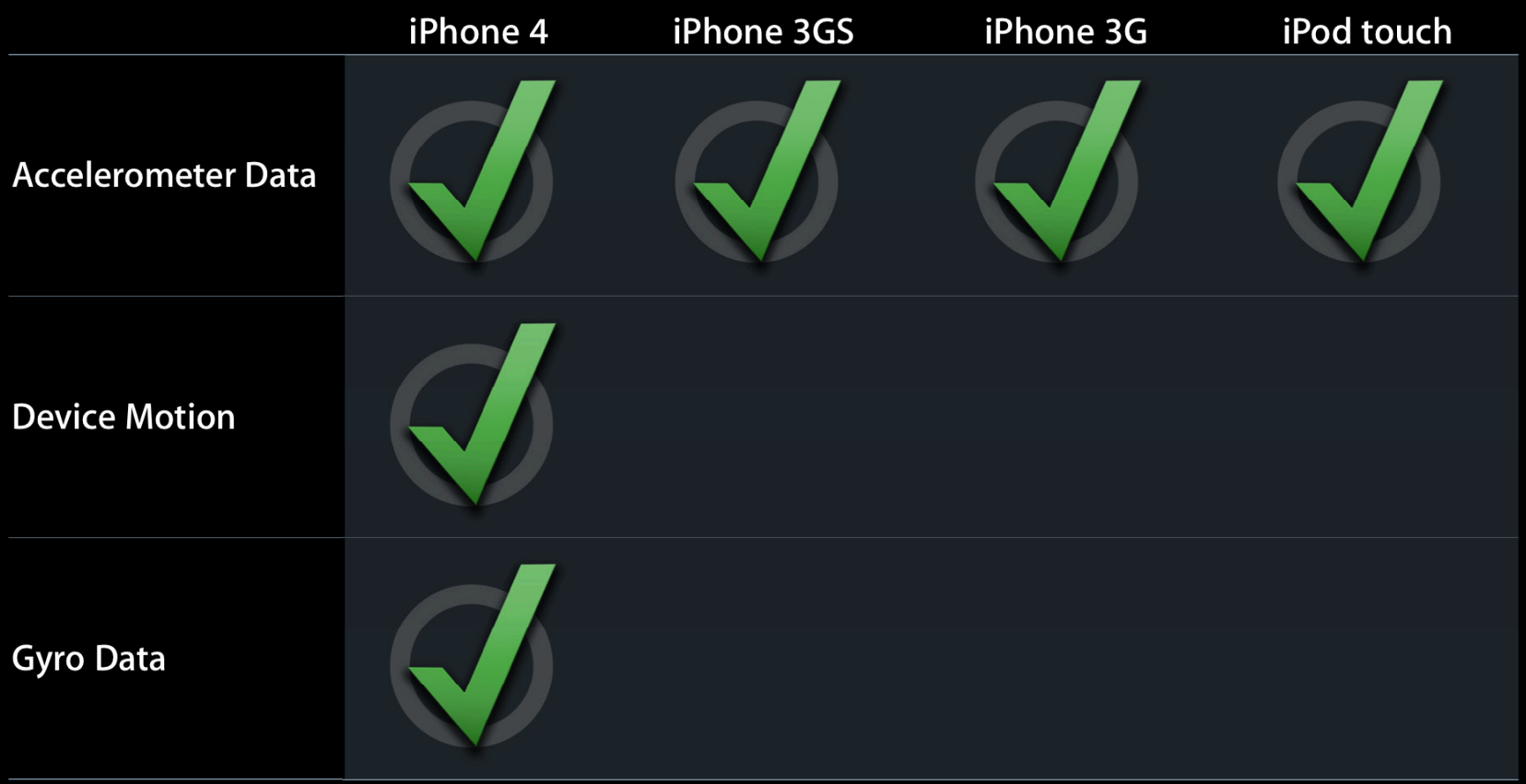
Core Motion

New framework in iOS 4

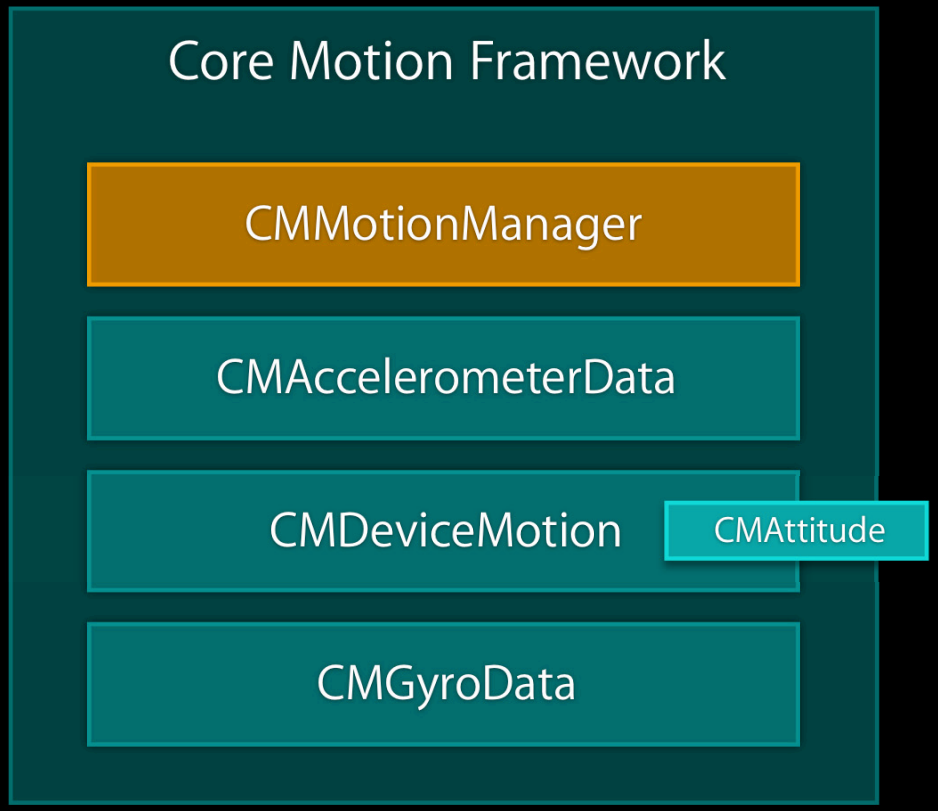
What Does Core Motion Provide?



Availability Matrix



Main Core Motion Objects



No Simulator Support

Retrieving Data

Push and pull

- Push
  - Must provide NSOperationQueue and block
- Pull
  - Periodically ask CMMotionManager for latest sample
  - Often done when view is updated

Retrieving Data

Push vs. pull tradeoffs

	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

Threading

- Core Motion creates its own thread to:
  - Handle raw data from sensors
  - Run device motion algorithms
- Pushing data:
  - Only your block will execute on your threads
- Pulling data:
  - Core Motion will never interrupt your threads

Outline for Using Core Motion

1. Setup
2. Retrieve data
3. Clean-up

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.isDeviceMotionAvailable) {
        // Fail gracefully
    }

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

    // Start updates
    // Note: We could call the following here instead:
    // [motionManager startDeviceMotionUpdatesToQueue:withHandler:]
    [motionManager startDeviceMotionUpdates];

}
```

Step 2: Retrieving Data

```
-(void) drawView:(id)sender
{

    CMDeviceMotion *newestDeviceMotion = motionManager.deviceMotion;

    // ...
}
```

Step 3: Cleaning Up

```
-(void) stopAnimation
{
    [motionManager stopDeviceMotionUpdates];
    [motionManager release];

    //...
}
```

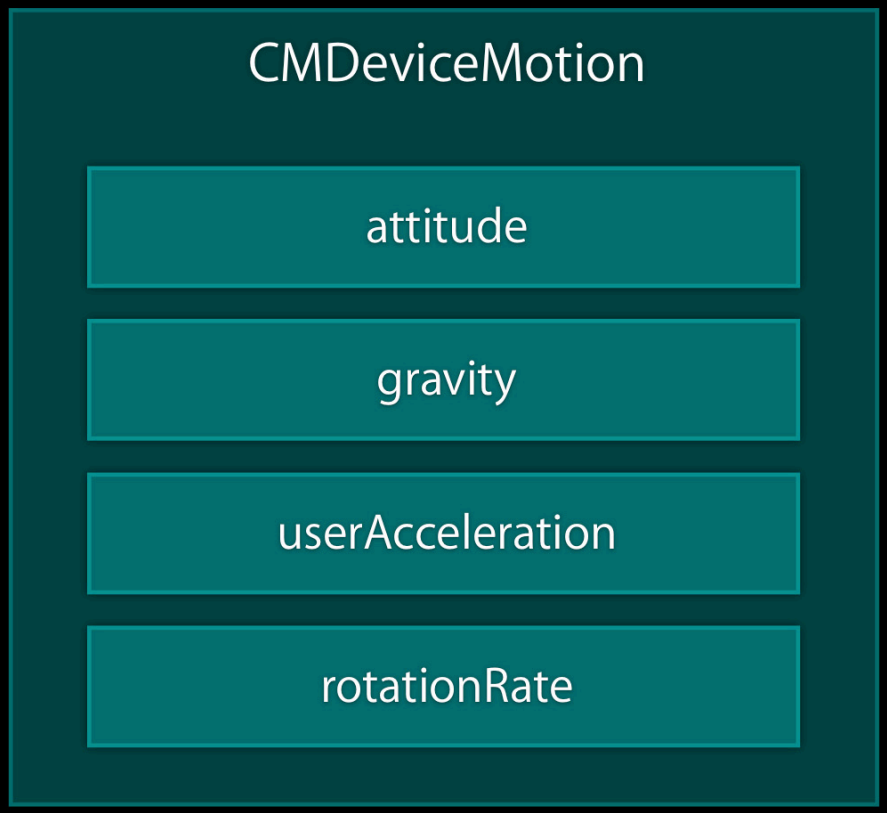
Using Core Motion

Summary

- Two methods to receive data:
  - Push
  - Pull
- Processing done on Core Motion's own thread
- Three steps to use Core Motion:
  - Setup
  - Retrieve data
  - Cleanup

Deep Dive into Device Motion

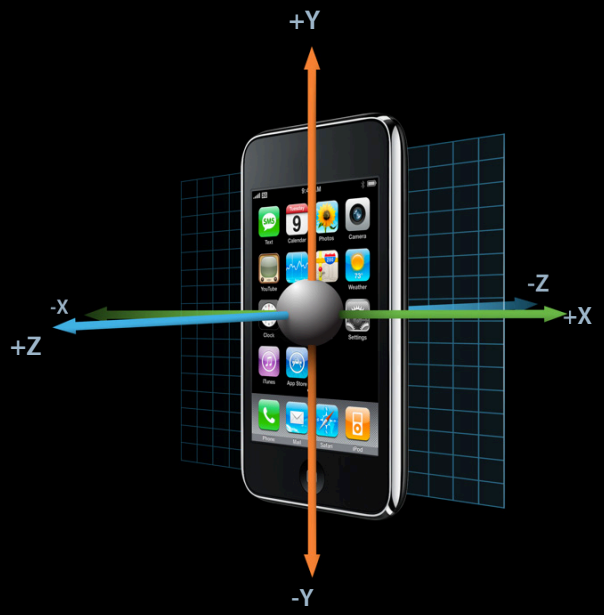
CMDeviceMotion Properties



Gravity and User Acceleration

```
@property(readonly, nonatomic) CMAcceleration gravity;
@property(readonly, nonatomic) CMAcceleration userAcceleration;
```

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



Low-Pass Filtering User Acceleration

```
static const double kFilterConst = 0.1;
```

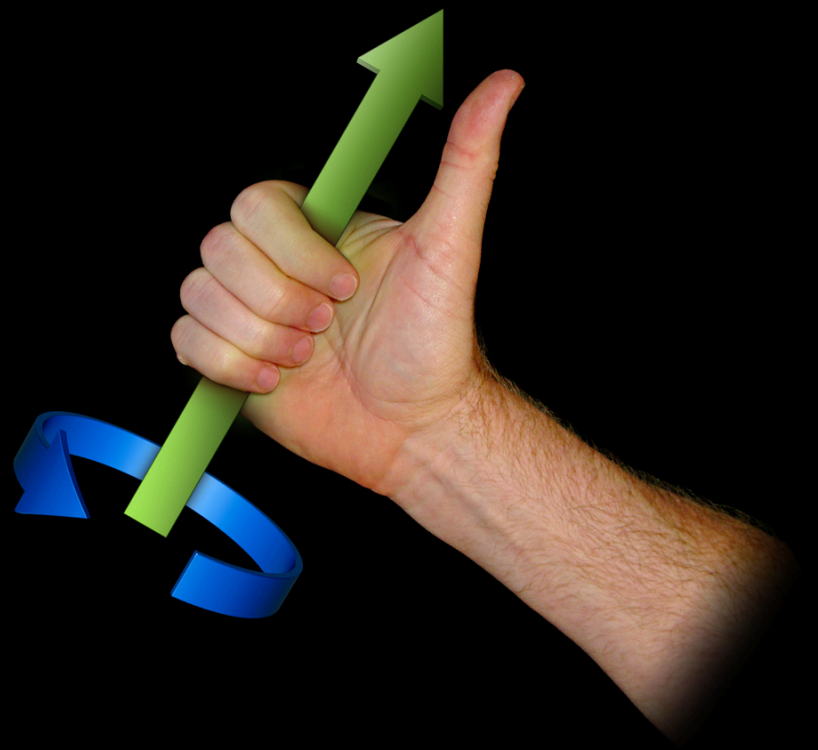
```
// motionManager is an instance of CMMotionManager
CMAcceleration accel = motionManager.deviceMotion.userAcceleration;
```

```
// userAccel is an instance of CMAcceleration
userAccel.x = userAccel.x*(1.0 - kFilterConst) + accel.x* kFilterConst;
userAccel.y = userAccel.y*(1.0 - kFilterConst) + accel.y* kFilterConst;
userAccel.z = userAccel.z*(1.0 - kFilterConst) + accel.z* kFilterConst;
```

Rotation Rate

```
@property(readonly, nonatomic) CMRotationRate rotationRate;
```

```
// Units are radians/second
typedef struct {
 double x;
 double y;
 double z;
} CMRotationRate;
```



How CMDeviceMotion's rotationRate Property Differ from CMGyroData?

bias!

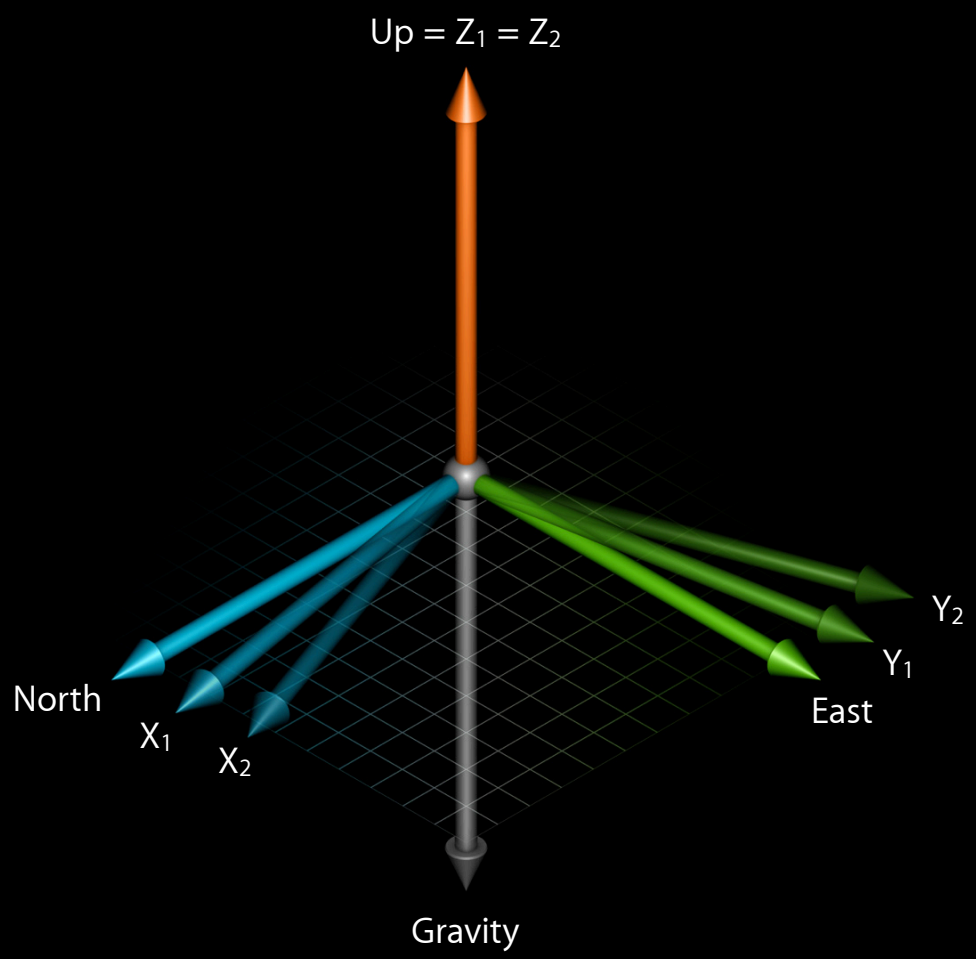
Attitude

```
@property(readonly, nonatomic) CMAttitude *attitude;
```

- Orientation of the device in 3D
- Ways to express:
  - Rotation matrix
  - Quaternion
  - Euler angles (pitch, roll, yaw)

Reference Frame

- Chosen when your app starts device motion updates
- Z axis is always vertical
  - Gravity is always [0, 0, -1]
- X and Y axes are both orthogonal to gravity



Example

```
CMDeviceMotion *deviceMotion = motionManager.deviceMotion;
```

```
CMRotationMatrix R = deviceMotion.attitude.rotationMatrix;
```

```
CMAcceleration gravityReference = {0.0, 0.0, -1.0};
```

```
// gravityDevice == deviceMotion.gravity
gravityDevice = multiplyMatrixAndVector(R, gravityReference);
```

$$\texttt{deviceMotion.gravity} = R \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

Changing Reference Frame

- Why
  - Provide comfortable “resting” orientation
- How

```
-[CMAttitude multiplyByInverseOfAttitude:]
```

Changing Reference Frame

Time

```
// Frame 1: Set Reference Frame

referenceAttitude = [motionManager.deviceMotion.attitude retain];
```

```
// Frame N

attitude = motionManager.deviceMotion.attitude;
[attitude multiplyByInverseOfAttitude: referenceAttitude];
```

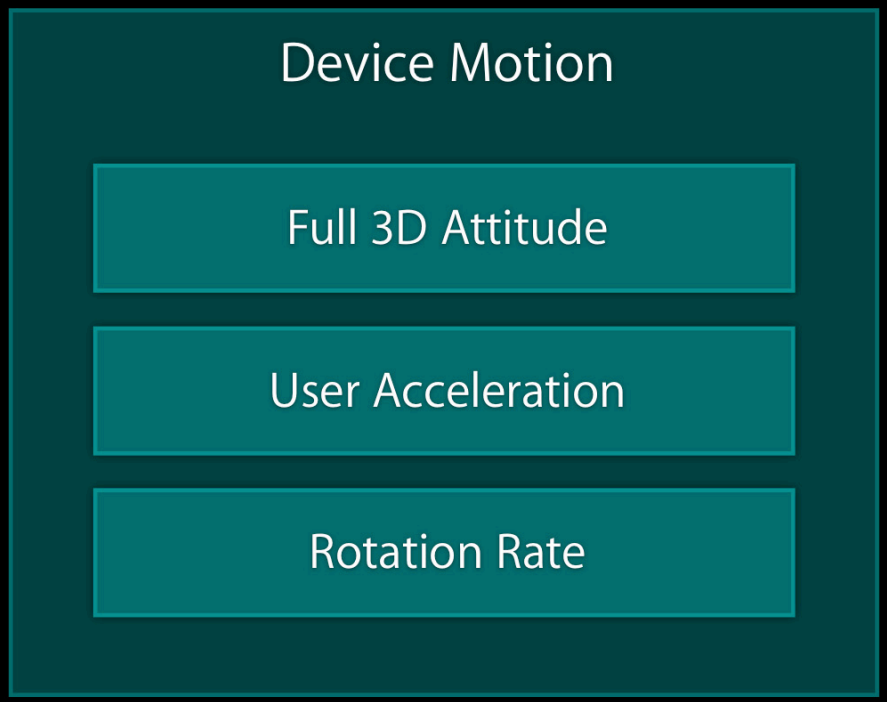
Demo

Patrick Piemonte
iPhone OS Software Engineer

Let's Code!

Wrapping Up

New Information



What Do We Do with This?

- GPS aiding
- Compass aiding

What Can You Do with This?

- Games
  - Simulations
  - Racing games
  - Boxing/fighting games
- Augmented reality
- 3D visualization
- Much, much more!

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

Game Design and Development for iPhone OS, Part 1 (Repeat)	Presidio Friday 9:00AM
Game Design and Development for iPhone OS, Part 2 (Repeat)	Presidio Friday 10:15AM
Introducing Blocks and Grand Central Dispatch on iPhone	Russian Hill Wednesday 11:30AM
OpenGL ES Overview for iPhone OS	Presidio Wednesday 2:00PM
Using Core Location in iOS 4	Presidio Wednesday 10:15AM

Labs

Core Motion Lab	Graphics and Media Lab D Thursday 11:30AM
Game Design for iPhone OS Lab	Graphics and Media Lab A Friday 11:30AM
OpenGL ES Lab	Graphics and Media Lab A Thursday 9:00AM

Q&A


WWDC10