

[Core Motion](#) / Getting raw accelerometer events

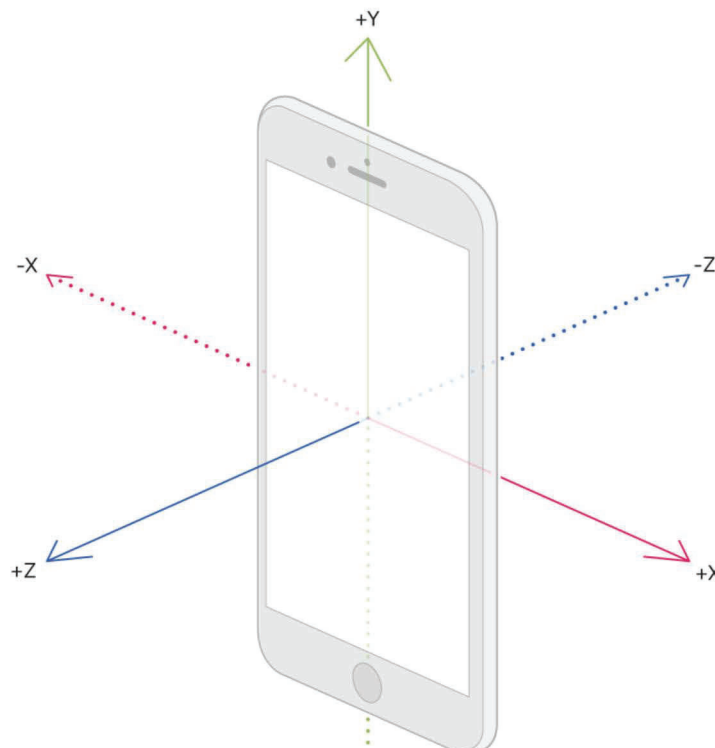
Article

Getting raw accelerometer events

Retrieve data from the onboard accelerometers.

Overview

An accelerometer measures changes in velocity along one axis. All iOS devices have a three-axis accelerometer, which delivers acceleration values in each of the three axes shown in the next illustration. The values reported by the accelerometers are measured in increments of the gravitational acceleration, with the value `1.0` representing an acceleration of 9.8 meters per second (per second) in the given direction. Acceleration values may be positive or negative depending on the direction of the acceleration.





You access the raw accelerometer data using the classes of the Core Motion framework. Specifically, the `CMMotionManager` class provides the interfaces for enabling the accelerometer hardware. When enabling the hardware, choose the interfaces that are best suited for your app. You can pull the accelerometer data only when you need it, or you can ask the framework to push updates to your app at regular intervals. Each technique involves different configuration steps and has a different use case.

Important

If your app relies on the presence of accelerometer hardware, configure the `UIRequiredDeviceCapabilities` key of its `Info.plist` file with the `accelerometer` value. For more information about the meaning of this key, see [Information Property List Key Reference](#).

For information about the coordinate axes of different device types, see [CMMotionManager](#).

Check for the availability of accelerometer data

Accelerometer data might be unavailable for a variety of reasons, so verify that the data is available before you try to obtain it. Check the value of the `isAccelerometerAvailable` property of `CMMotionManager` and make sure it's `true`. If it's `false`, starting updates doesn't deliver any data to your app.

Important

In visionOS, accelerometer data is available only when your app has an open immersive space. For more information, see [ImmersiveSpace](#).

Get accelerometer data only when you need it

For apps that process accelerometer data on their own schedule, such as games, use the `startAccelerometerUpdates()` method of `CMMotionManager` to start the delivery of accelerometer data. When you call this method, the system enables the accelerometer hardware and begins updating the `accelerometerData` property of your `CMMotionManager` object. However, the system does not notify you when it updates that property. You must explicitly check the value of the property when you need the accelerometer data.

Before you start the delivery of accelerometer updates, specify an update frequency by assigning a value to the `accelerometerUpdateInterval` property. The maximum frequency at which you can request updates is hardware-dependent but is usually at least 100 Hz. If you request a frequency that is greater than what the hardware supports, Core Motion uses the supported maximum instead.

The example below shows a method that configures accelerometer updates to occur 50 times per second. The method then configures a timer to fetch those updates at the same frequency and do something with the data. You could configure the timer to fire at a lower frequency, but doing so would waste power by causing the hardware to generate more updates than were actually used.

```
let motion = CMMotionManager()

func startAccelerometers() {
    // Make sure the accelerometer hardware is available.
    if self.motion.isAccelerometerAvailable {
        self.motion.accelerometerUpdateInterval = 1.0 / 50.0 // 50 Hz
        self.motion.startAccelerometerUpdates()

        // Configure a timer to fetch the data.
        self.timer = Timer(fire: Date(), interval: (1.0/50.0),
                           repeats: true, block: { (timer) in
            // Get the accelerometer data.
            if let data = self.motion.accelerometerData {
                let x = data.acceleration.x
                let y = data.acceleration.y
                let z = data.acceleration.z

                // Use the accelerometer data in your app.
            }
        })

        // Add the timer to the current run loop.
        RunLoop.current.add(self.timer!, forMode: .defaultRunLoopMode)
    }
}
```

Process a steady stream of accelerometer data

When you want to capture all of the incoming accelerometer data, perhaps so you can analyze it for movement patterns, use the `startAccelerometerUpdates(to:withHandler:)` method of `CMMotionManager`. This method pushes each new set of accelerometer values to your app by executing your handler block on the specified queue. The queueing of these blocks ensures that your app receives all of the accelerometer data, even if your app becomes busy and is unable to process updates for a brief period of time.

Before you start the delivery of accelerometer updates, specify an update frequency by assigning a value to the `accelerometerUpdateInterval` property. The maximum frequency at which you can request updates is hardware-dependent but is usually at least 100 Hz. If you request a frequency that is greater than what the hardware supports, Core Motion uses the supported maximum instead.

The following example shows a method from the `MotionGraphs` sample code project, which you can examine for more context. The app displays a real-time graph of accelerometer data. The user configures the update frequency for the accelerometers using a slider, the changing of which results in a call to the `startUpdatesWithSliderValue:` method shown in the example. This method restarts the accelerometer updates with the new frequency. Each time a new sample is received, the specified block is queued on the main thread. That block updates the app's graph view and labels with the new accelerometer values.

```

static const NSTimeInterval accelerometerMin = 0.01;
- (void)startUpdatesWithSliderValue:(int)sliderValue {
    // Determine the update interval.
    NSTimeInterval delta = 0.005;
    NSTimeInterval updateInterval = accelerometerMin + delta * sliderValue;
    // Create a CMMotionManager object.
    CMMotionManager *mManager = [[APLAppDelegate *)
        [[UIApplication sharedApplication] delegate] sharedManager];
    APLAccelerometerGraphViewController * __weak weakSelf = self;
    // Check whether the accelerometer is available.
    if ([mManager isAccelerometerAvailable] == YES) {
        // Assign the update interval to the motion manager.
        [mManager setAccelerometerUpdateInterval:updateInterval];
        [mManager startAccelerometerUpdatesToQueue:[NSOperationQueue mainQueue]
            withHandler:^(CMAccelerometerData *accelerometerData, NSError *error)
        [weakSelf.graphView addX:accelerometerData.acceleration.x
            y:accelerometerData.acceleration.y
            z:accelerometerData.acceleration.z];
        [weakSelf setLabelValueX:accelerometerData.acceleration.x
            y:accelerometerData.acceleration.y
            z:accelerometerData.acceleration.z];
    }
}
self.updateIntervalLabel.text = [NSString stringWithFormat:@"%f", updateInterval]
}
- (void)stopUpdates {
    CMMotionManager *mManager = [[APLAppDelegate *)
        [[UIApplication sharedApplication] delegate] sharedManager];
    if ([mManager isAccelerometerActive] == YES) {
        [mManager stopAccelerometerUpdates];
    }
}
}

```

See Also

Accelerometers

`class CMAccelerometerData`

A data sample from the device's three accelerometers.

`class CMRecordedAccelerometerData`

A single piece of accelerometer data that was recorded by the device.

`class CMSensorRecorder`

An object that gathers and retrieves accelerometer data from a device.

`class CMSensorDataList`

A list of the accelerometer data recorded by the system.