Core Motion

Core Motion

Core Motion is a framework that allows your application to receive motion data from device hardware.

For an iOS developer, this means you can create applications that can observe and respond to the motion and orientation of an iOS device.

Important note:

You can only test or use the functionality of Core Motion on an actual device. The simulator does not have any facilities for reproducing physical motion for your app.

Hardware Elements of Core Motion

Accelerometer
- Measures acceleration in all three dimensions

Gyroscope
- Calculates orientation and rotation in all three dimensions

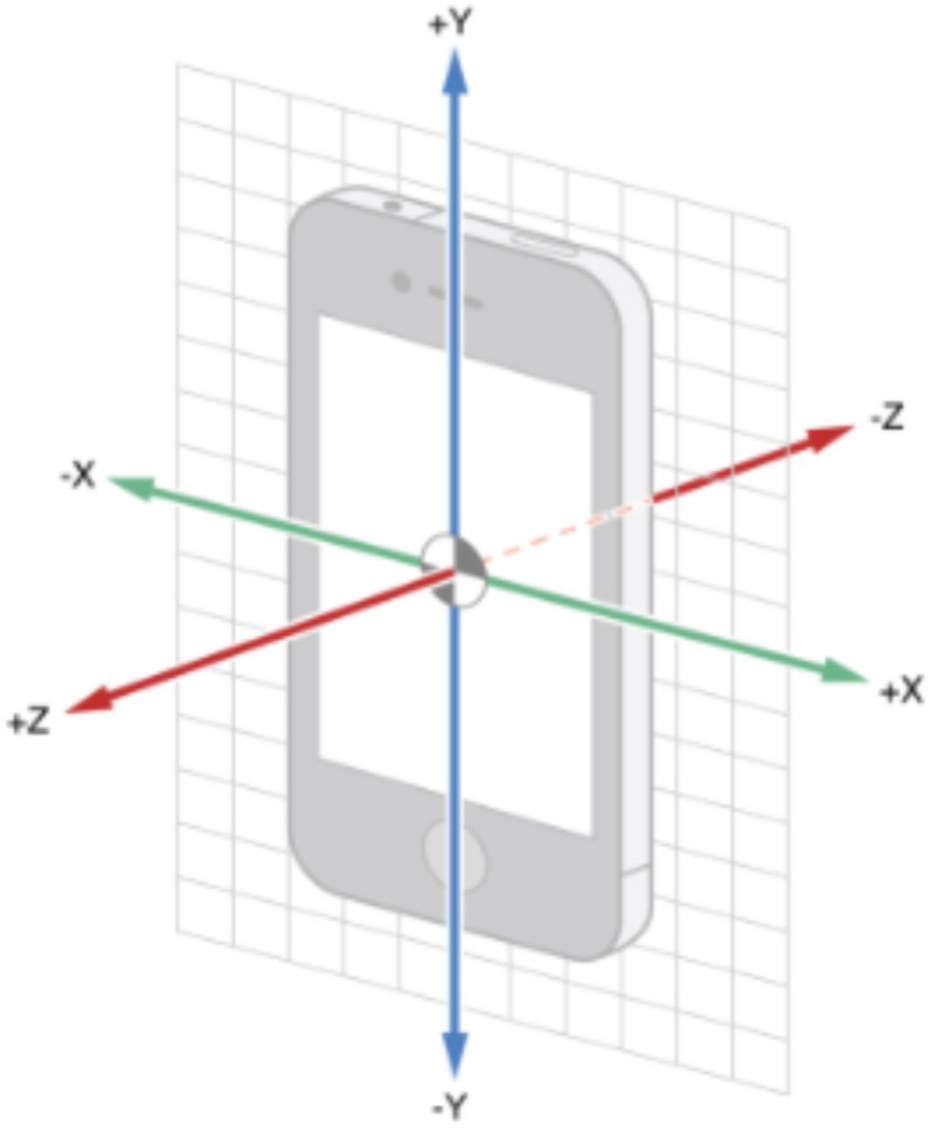
Magnetometer
- Measures magnetic forces

Coordinate System

+x is towards the right of the screen

+y is towards the top of the screen

+z is towards the user when the screen is faceup



CMDeviceMotion

A `CMDeviceMotion` object contains the following objects as properties:

- **`attitude:`** `CMAttitude`
  - Returns the orientation of the device
  - Can access the data in any of 3 representations
- **`rotationRate:`** `CMRotationRate`
  - Returns the rotation rate of the device for devices with a gyro
  - x, y, z values in radians per second
- **`gravity:`** `CMAcceleration`
  - Returns the gravity vector expressed in the device's reference frame
  - x, y, z values in g's (gravitational force)
- **`userAcceleration:`** `CMAcceleration`
  - Returns the acceleration that the user is giving to the device
  - x, y, z values in g's (gravitational force)
- **`magneticField:`** `CMCalibratedMagneticField`
  - Returns the magnetic field vector with respect to the device for devices with a magnetometer

`CMAttitude`

`CMAttitude` contains three different representations of the device's orientation:

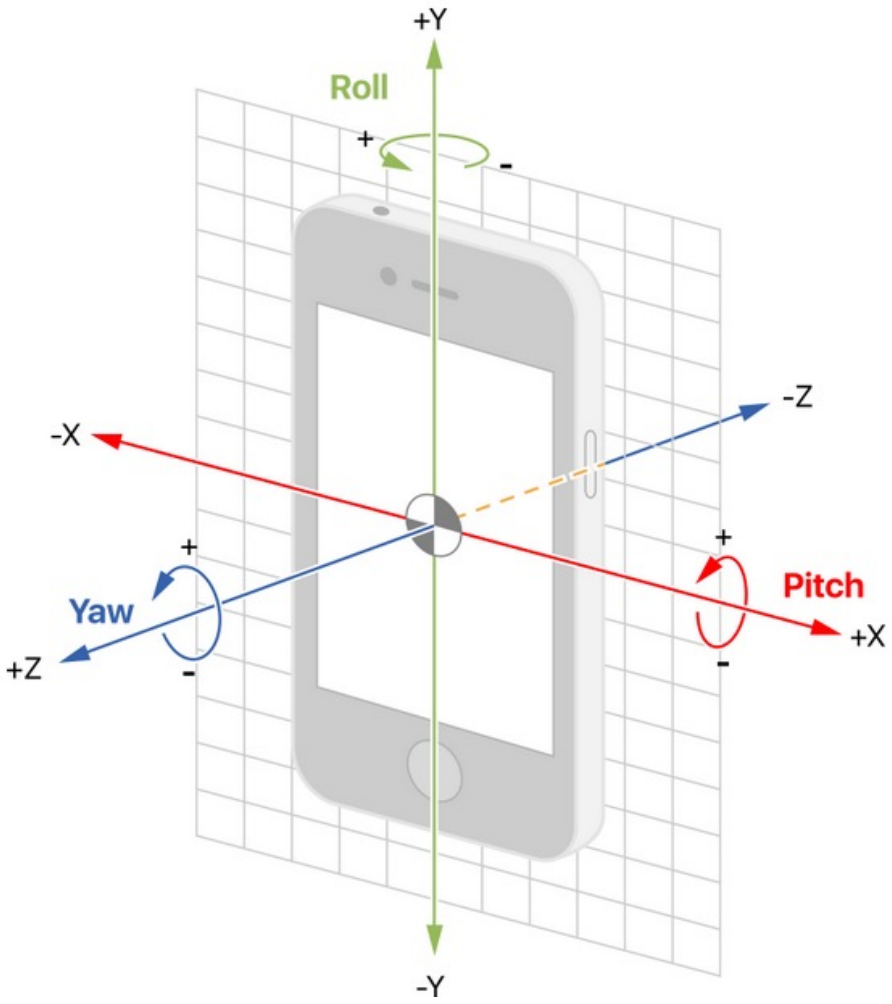
- Euler angles (pitch, roll, yaw)
- Rotation matrices
- Quaternions

Each of these is in relation to a given reference frame.

Euler Angles

Euler Angles are the most readily understood of the 3 representations, as they simply describe rotation around each of the axes.

- **Pitch** - is rotation around the **x-axis**, increasing as the device tilts toward you, decreasing as it tilts away
- **Roll** - is rotation around the **y-axis**, decreasing as the device rotates to the left, increasing to the right
- **Yaw** - is rotation around the **z-axis**, decreasing clockwise, increasing counter-clockwise



CMMotionManager

`CMMotionManager` provides a consistent interface for each of the four motion data types:

- Attitude (rotation)
- Acceleration
- Gravity
- Magnetic Field

Although you can access data for each of these motion types individually, it's simplest to create a `CMMotionManager` instance to access all of the above.

If `deviceMotion` is a `CMDeviceMotion` object:

```
deviceMotion.gravity.x
deviceMotion.gravity.y
deviceMotion.gravity.z

deviceMotion.userAcceleration.x
deviceMotion.userAcceleration.y
deviceMotion.userAcceleration.z

deviceMotion.attitude.pitch
deviceMotion.attitude.roll
deviceMotion.attitude.yaw

deviceMotion.magneticField.field.x
deviceMotion.magneticField.field.y
deviceMotion.magneticField.field.z
```

Using Core Motion in Your App

1. Create a motion manager:

In your `ViewController` class:

```
let motionManager = CMMotionManager()
```

2. Start receiving updates at the desired frequency:

```
override func viewDidLoad() {
    super.viewDidLoad()

    motionManager.deviceMotionUpdateInterval = 0.1

    motionManager.startDeviceMotionUpdates(to:
                          OperationQueue.current!) {
        (deviceMotion, error) -> Void in

        if(error == nil) {    // you write these methods
            self.handleUpdate(deviceMotion: deviceMotion!)
        } else {
            self.handleError()
        }
    }
}
```

Using Core Motion in Your App

3. Write code specifying what you want to happen at each update

```
func handleUpdate(deviceMotion:CMDeviceMotion) {

    let acceleration = deviceMotion.userAcceleration
    let xAcc = acceleration.x
    let yAcc = acceleration.y
    let zAcc = acceleration.z

    print("Acceleration in the x direction: \(xAcc) ")
    print("Acceleration in the y direction: \(yAcc) ")
    print("Acceleration in the z direction: \(zAcc) ")
}

func handleError() {
    print("An error occurred")
```