移动平台开发技术

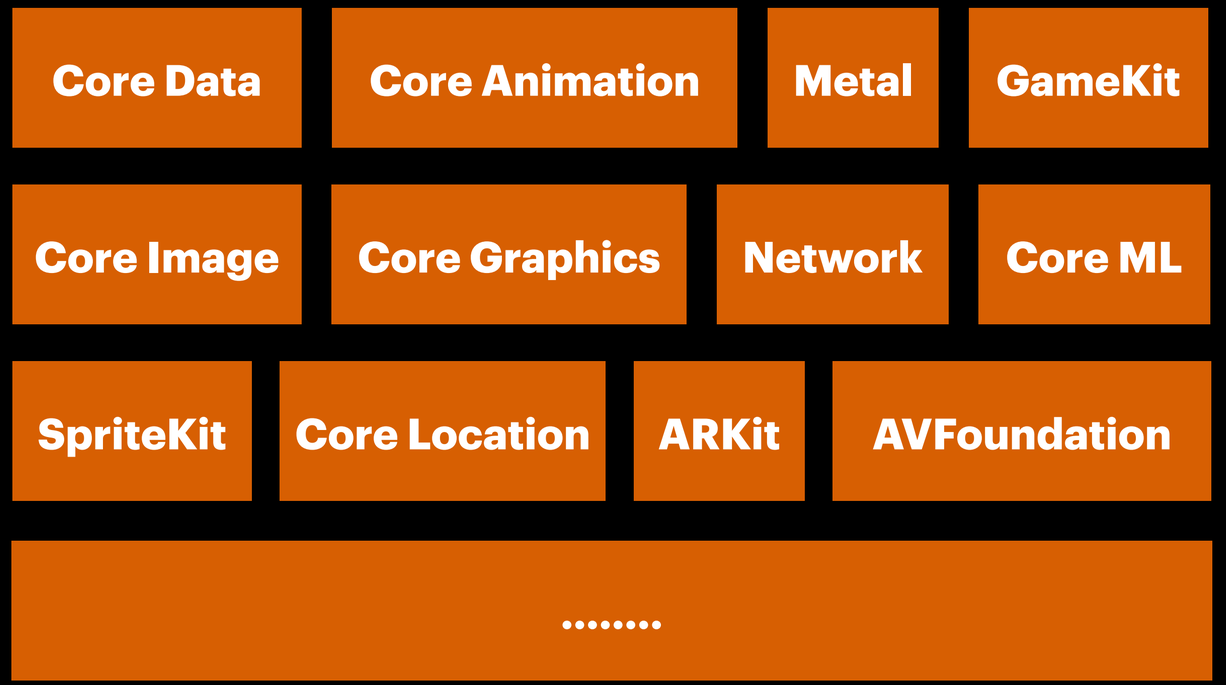
iOS开发中的传感器

章国锋　2025春夏学期

Frameworks

Frameworks

- 苹果为开发者提供了大量方便快捷的框架
- 在开发时候可以直接调用，缩短开发时间
- 本次课程我们重点介绍：
  - CoreMotion
  - CoreLocation
  - AVFoundation
  - Core Image



浙江大学
ZHEJIANG UNIVERSITY

IMU基础知识Core Motion

Core Location

AVFoundation 与 Core Image

浙江大學
ZHEJIANG UNIVERSITY

IMU基础知识Core Motion

Core Location

AVFoundation 与 Core Image

IMU: 惯性测量单元

IMU: Inertial Measurement Unit

浙江大学
ZHEJIANG UNIVERSITY

惯性测量单元 (Inertial Measurement Unit, IMU) 通常包含三个互相垂直的速率陀螺仪 (gyroscope) 以及三个互相垂直的加速度计(accelerometer)，分别用于测量角速度和线加速度

IMU: 惯性测量单元

IMU: Inertial Measurement Unit

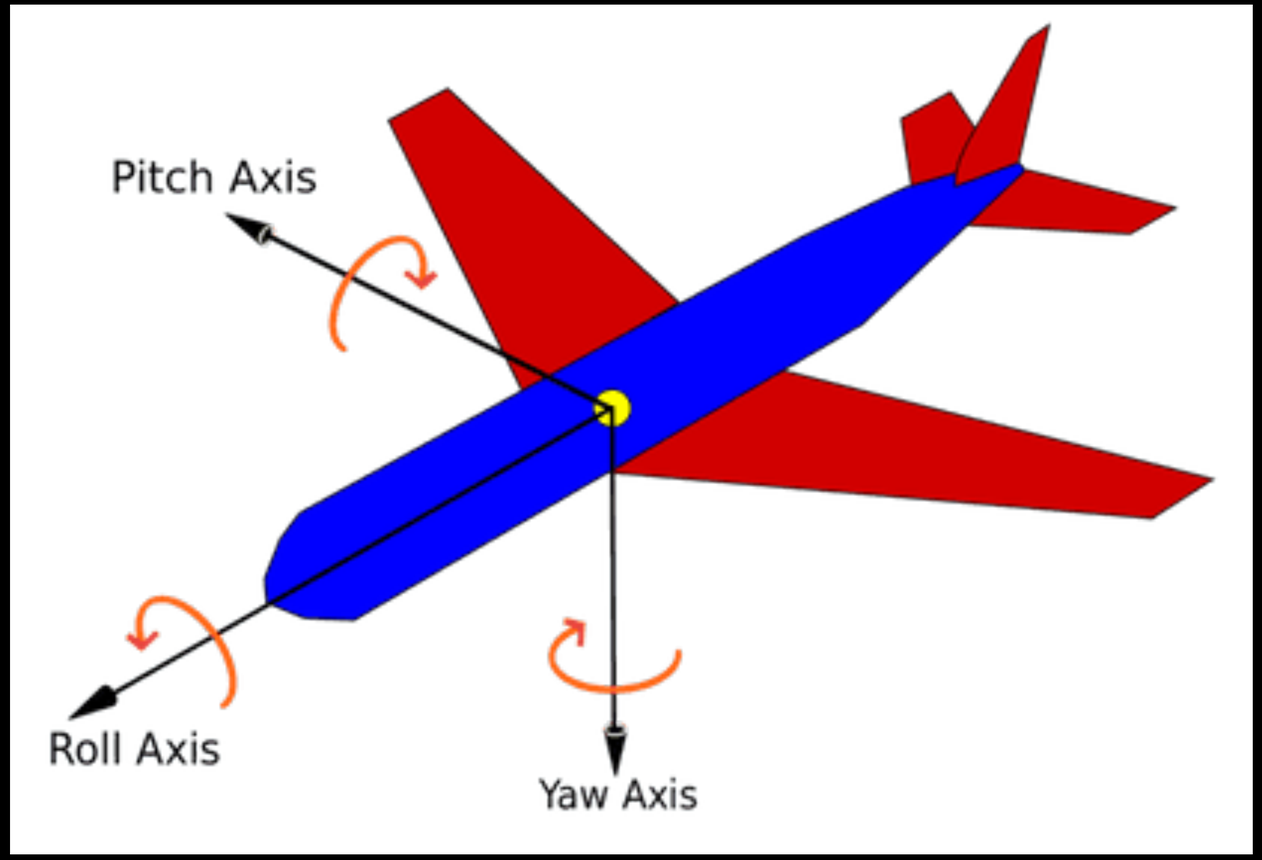
浙江大学 ZHEJIANG UNIVERSITY

http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

IMU: 惯性测量单元

IMU: Inertial Measurement Unit

- 使用 IMU 的信息源，加上传感器融合算法，就可以估计系统的姿态，通常用欧拉角 pitch，roll，yaw 表示



http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

陀螺仪

Gyroscope

- 机械式陀螺仪由一个旋转轮，两个互相垂直的**万象架（gimbal）**组成。核心原理是**角动量守恒**。

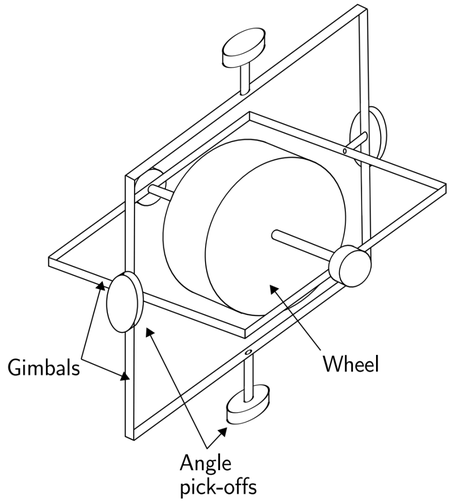
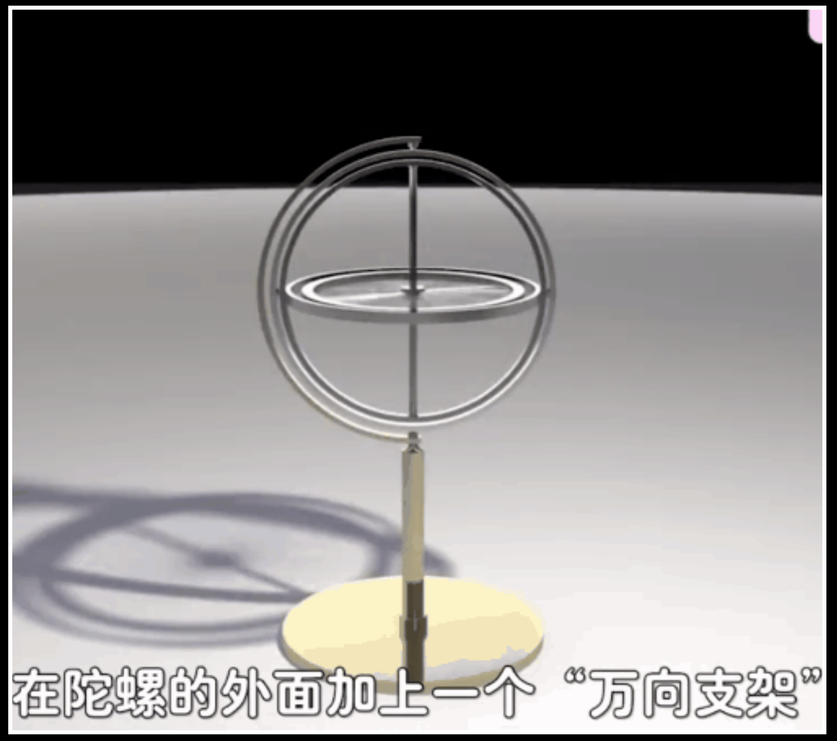


Figure 5: A conventional mechanical gyroscope (source: [1]).



https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

陀螺仪：MEMS微机电系统

Gyroscope：Micro-Electro Mechanical System

浙江大学 ZHEJIANG UNIVERSITY

- 目前常用于电子设备中的陀螺仪是微机电式的系统。其原理是**科里奥利效应 (Coriolis Effect)**
- 某个质量为 m 的物体水平方向有个速度为 υ 的运动，同时又存在一个速度为 ω 的角速度。因此，它就会在垂直方向受到一个科里奥利力 $Fc = -2m\,(\,\omega \times \upsilon\,)$

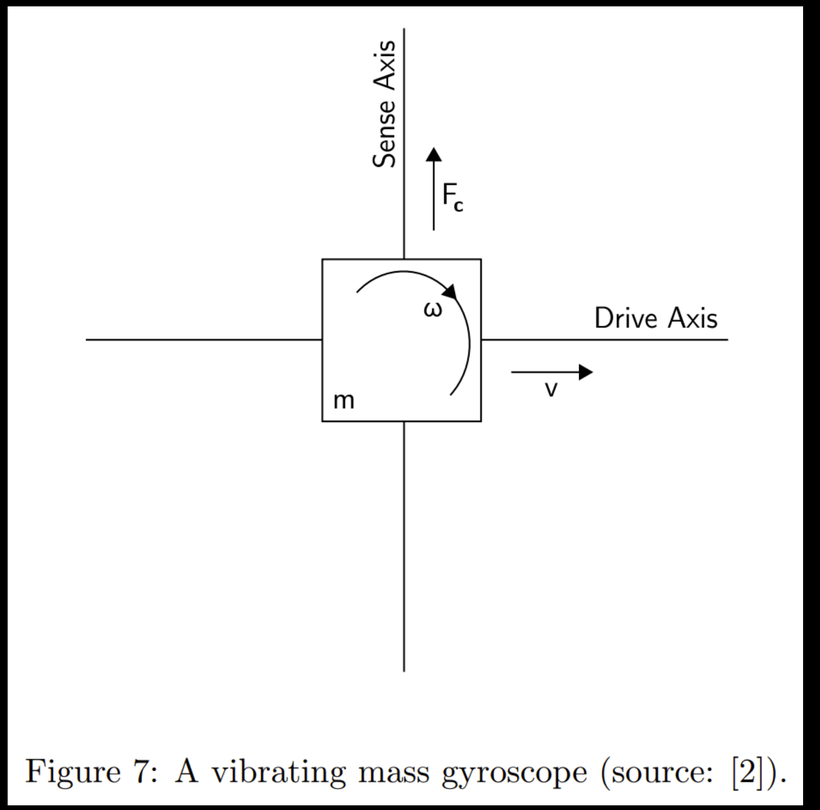


Figure 7: A vibrating mass gyroscope (source: [2]).

https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

陀螺仪：MEMS微机电系统

Gyroscope：Micro-Electro Mechanical System

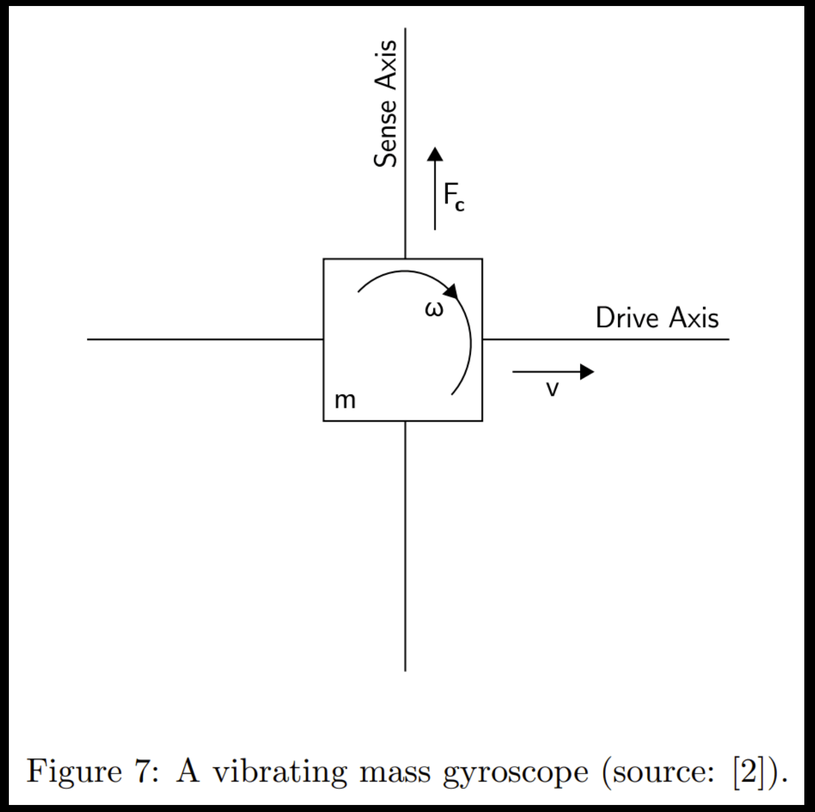
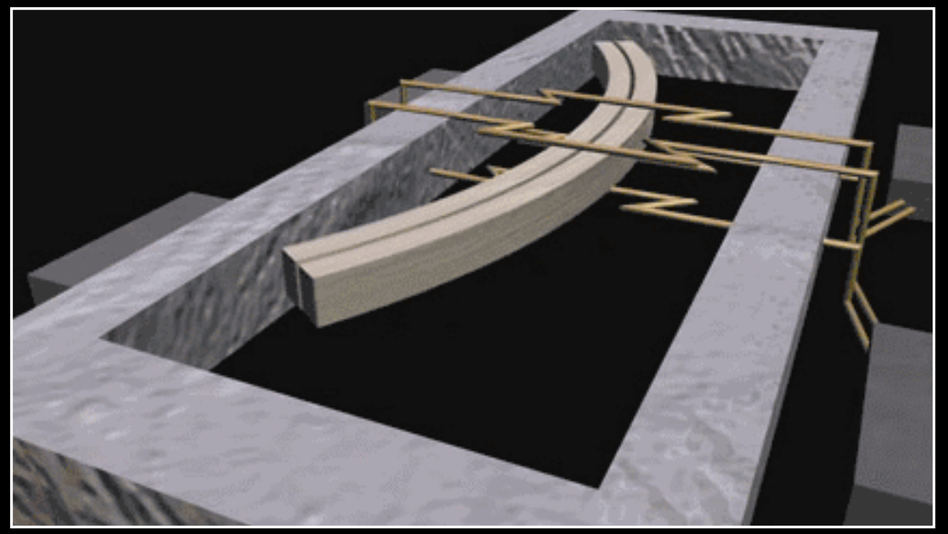


Figure 7: A vibrating mass gyroscope (source: [2]).

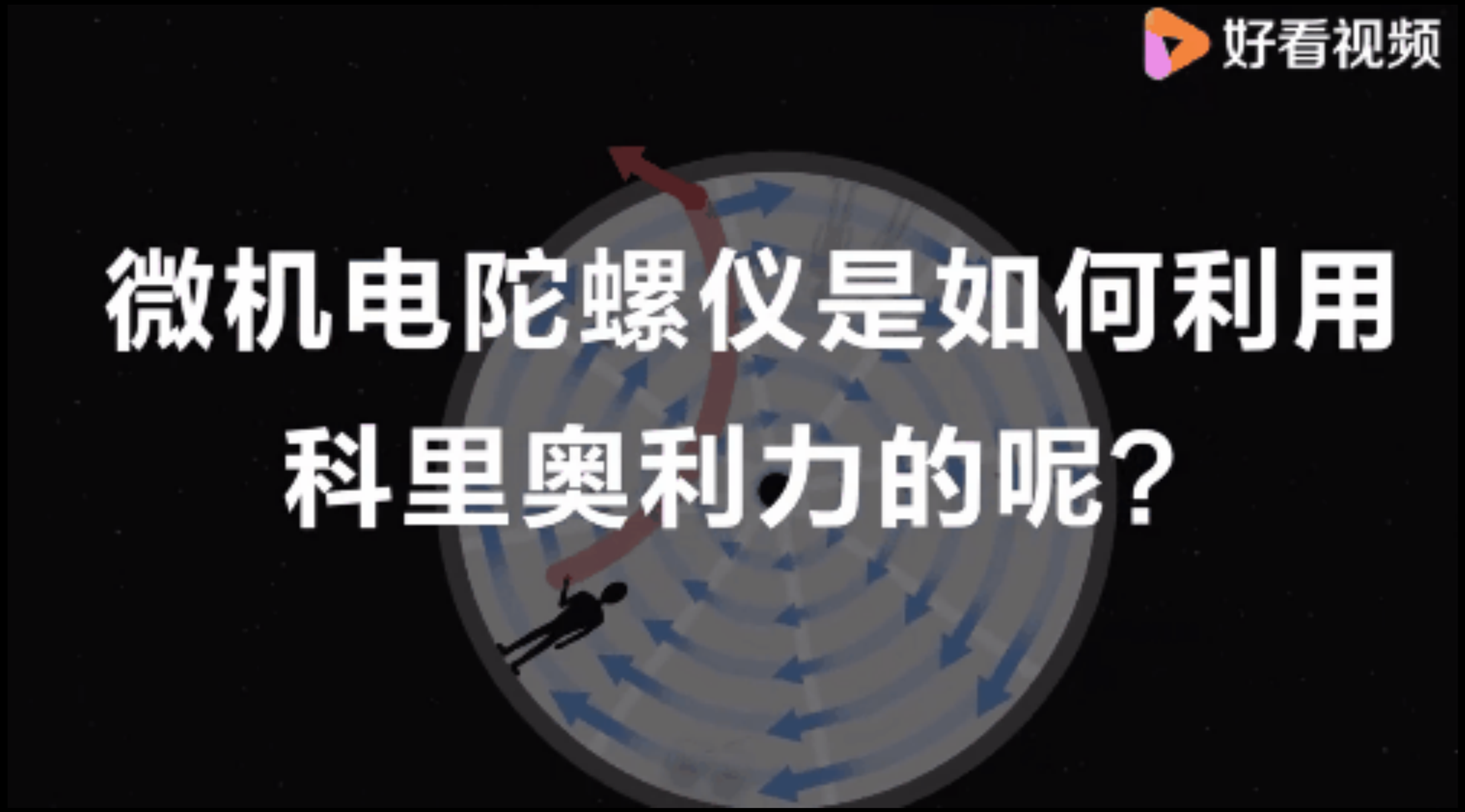
- 方法概述：已知 m、v、Fc，求解 ω 角速度
- 怎么测量科氏力 Fc？可以通过微机械结构电容的变化来间接测量科里奥利力



https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

陀螺仪：MEMS微机电系统

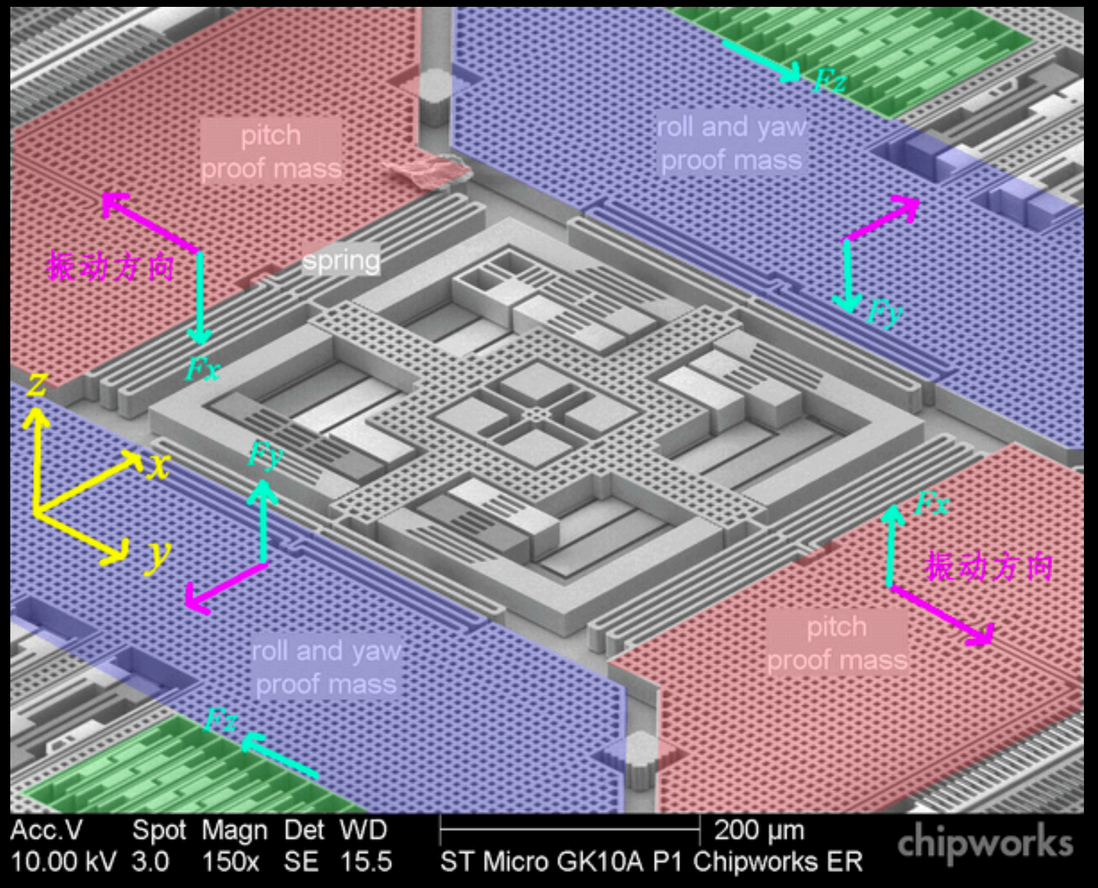
Gyroscope：Micro-Electro Mechanical System



陀螺仪：MEMS微机电系统

Gyroscope：Micro-Electro Mechanical System

- 使用压电效应产生一个震动机构
- 当物体旋转的时候，科氏力会引起电容电容变化
- 代入科氏力公式，可以算出角速度



http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

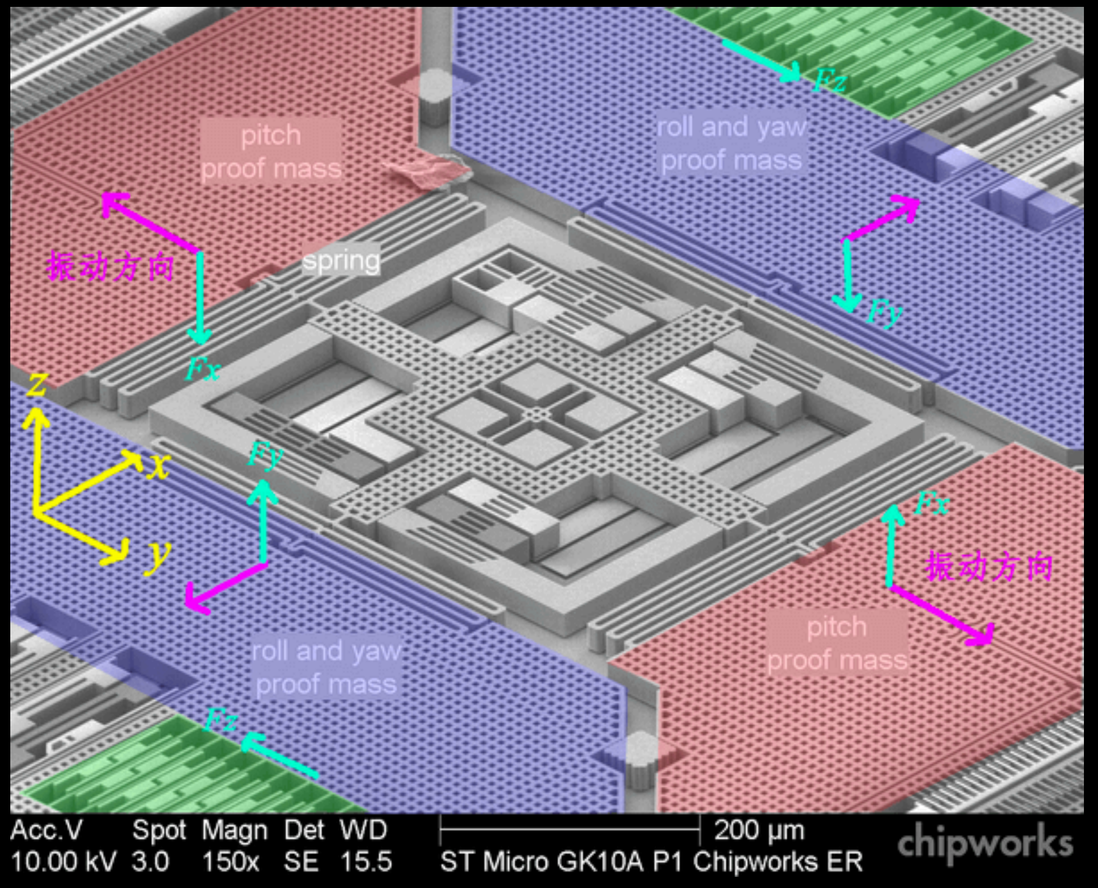
陀螺仪：MEMS微机电系统

Gyroscope：Micro-Electro Mechanical System

浙江大学 ZHEJIANG UNIVERSITY

MEMS陀螺仪和传统陀螺仪相比的优点：

- 小尺寸、质量轻
- 结构坚固、不易损坏
- 低功耗
- 启动时间短
- 成本低
- 高可靠性



http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

加速度计

Accelerometer

常见的加速度计有两种，其原理都是通过牛顿第二定律，在已知 F 和 m的情况下求 a

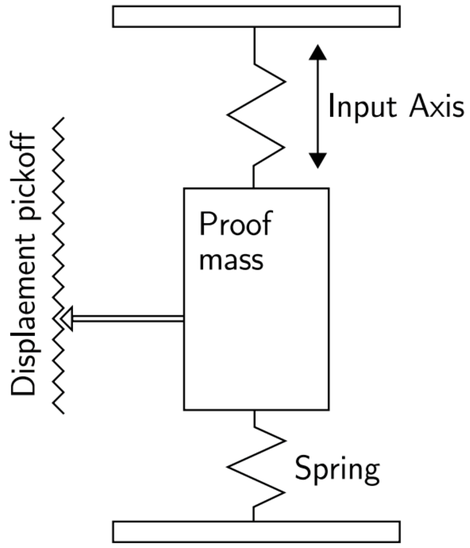


Figure 8: A mechanical accelerometer (source: [1]).

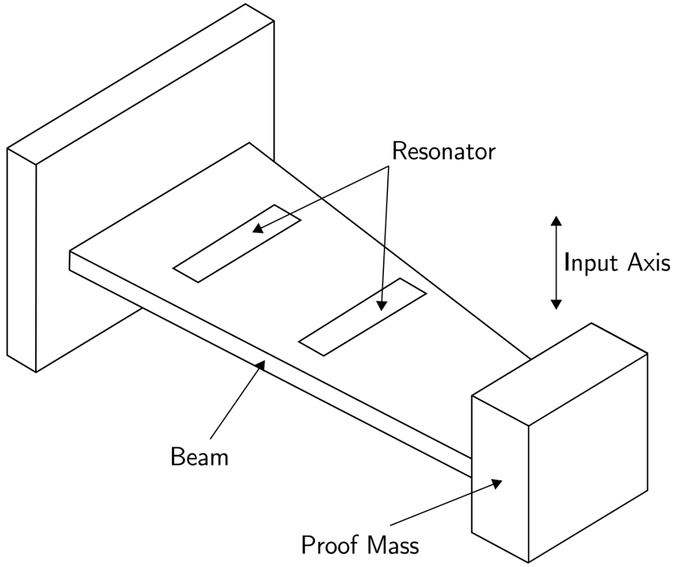


Figure 9: A surface acoustic wave accelerometer (source: [1]).

机械式加速度计和固态加速度计

https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

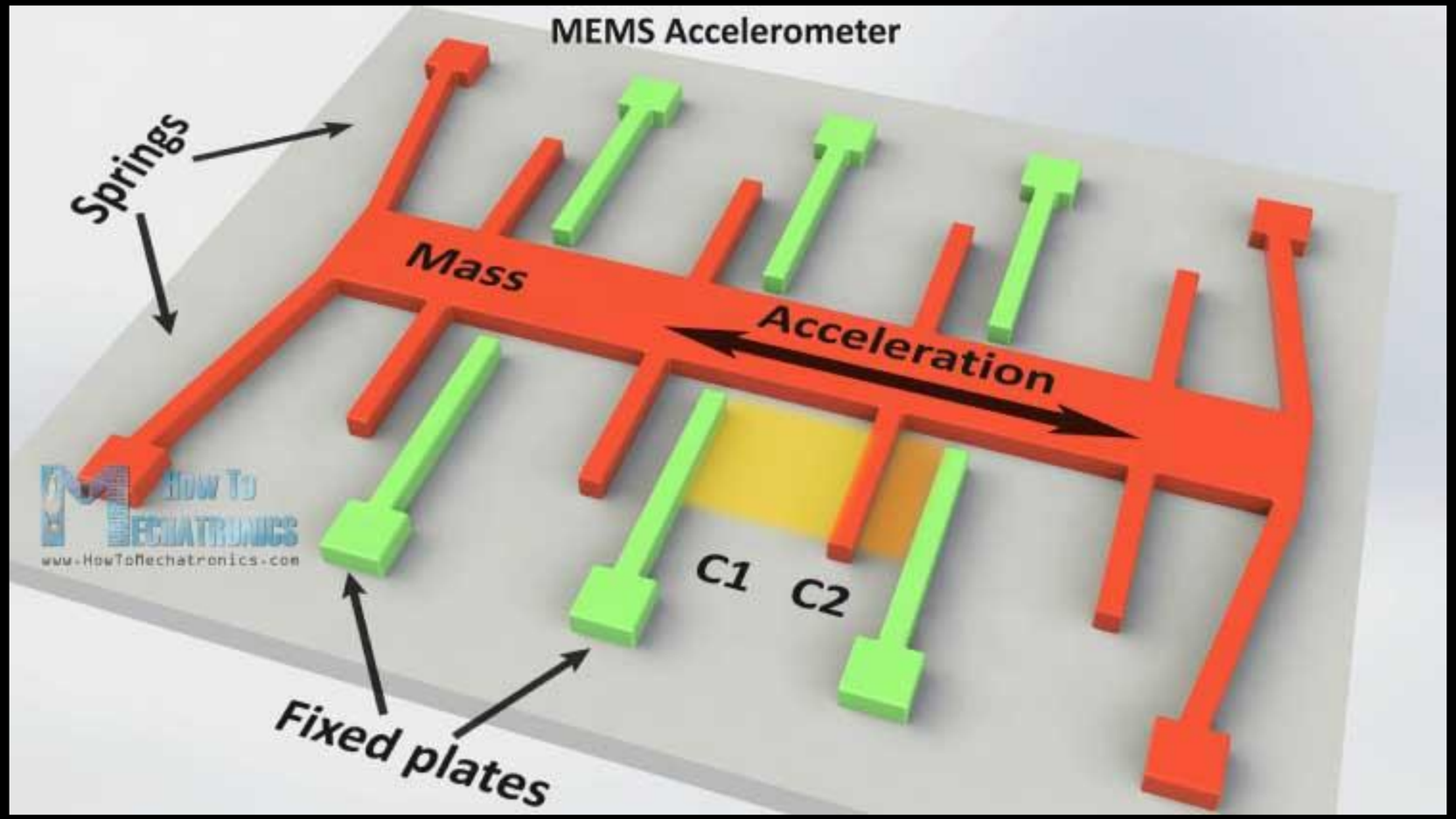
加速度计

Accelerometer

现代电子设备中

一般采用 MEMS

实现加速度计



MEMS：误差类别

MEMS：Error Type

浙江大学 ZHEJIANG UNIVERSITY

MEMS器件产生误差的原因：

- 常量偏置(Constant Bias)：可以看作是一种非常低频的常量系统误差，有时候也会被称为零偏
- 热机械白噪声/角度随机游走(Thermo-Mechanical White Noise/Angle Random Walk)：受到热机械噪声的影响，频率远比传感器的采样频率要高，会给积分结果引入零均值随机游走误差。陀螺仪中，角度随机游走噪声的标准差与时间的平方根成正比；加速度计中，速度随机游走噪声的标准差与时间的 3/2 次方成正比

http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

MEMS：误差类别

MEMS：Error Type

浙江大学 ZHEJIANG UNIVERSITY

其他MEMS器件产生误差的原因：

- 闪变噪声/偏置稳定性(Flicker Noise/Bias Stability)：由于电子器件的闪变噪声(Flicker Noise)影响，MEMS的偏置也不会一直不变，而是随时间游走。因此，偏置稳定性就是用于描述某个设备在一段时间内(比如100s)偏置的变化情况。
- 温度影响(Temperature Effects)
- 校正误差(Calibration Errors)

http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

MEMS：误差类别

MEMS：Error Type

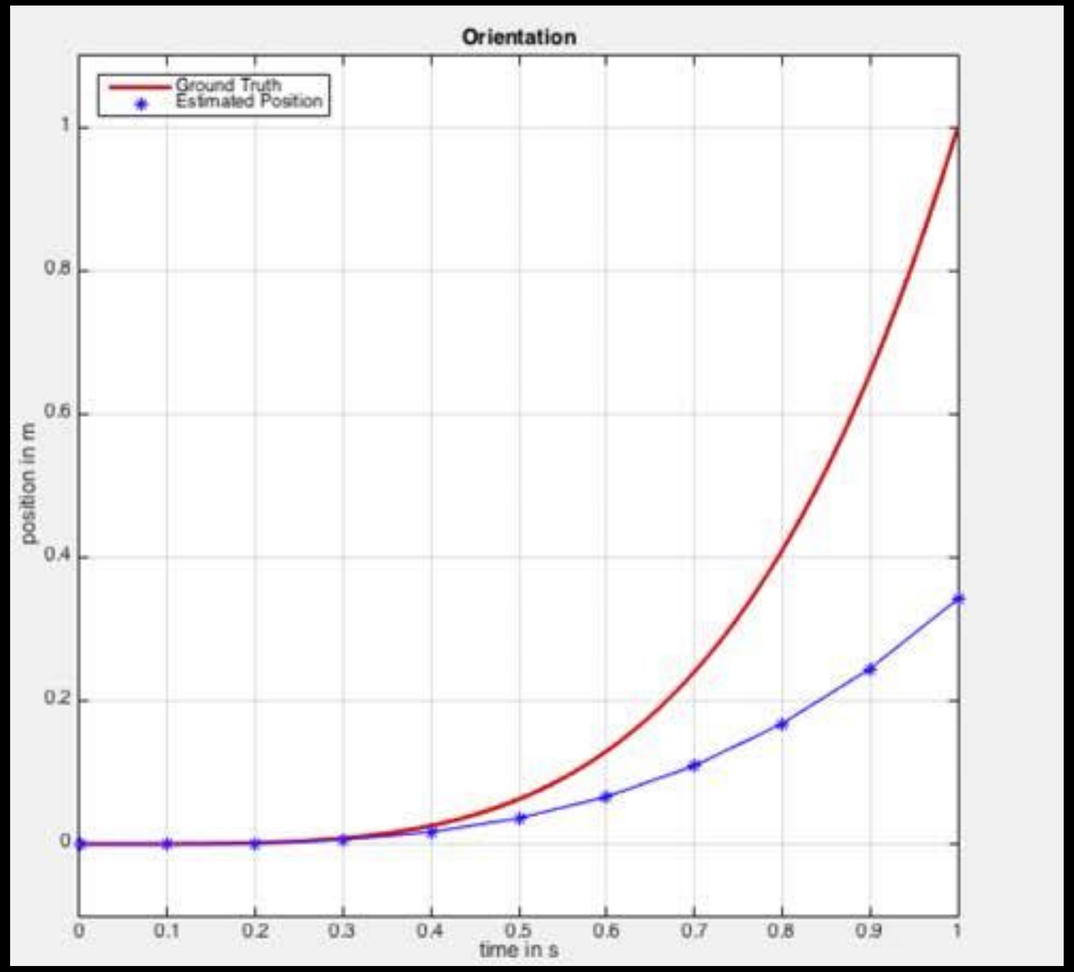
总结：对于MEMS而言，最主要的就是随机游走噪声(由于未补偿的温度波动引起)和未校正偏置(由于初始偏置估计误差引起)。以陀螺仪为例，这两类误差分别是：

- 静止系统误差，不随时间变化，但对角度积分的影响随时间线性增长；
- 随机误差，随时间变化，可以简化成高斯噪声，对角度积分的影响与时间的平方根成比例。

http://zhaoxuhui.top/blog/2022/02/11/basic-notes-on-imu-and-inertial-navigation.html

MEMS：误差类别

MEMS： Error Type



积分会导致随着时间推

移累计误差越来越大

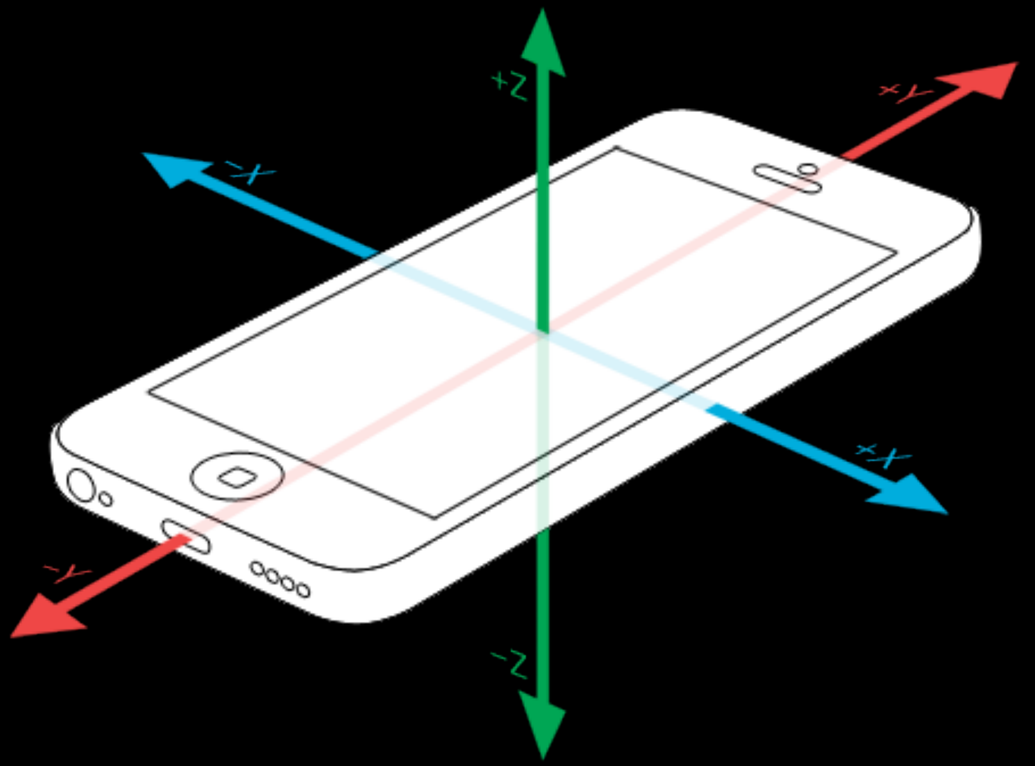
使用IMU进行姿态估计

Pose Estimation using IMU

浙江大学 ZHEJIANG UNIVERSITY

只使用单个传感器进行姿态估计：

- 使用陀螺仪：对角速度积分可以计算得到姿态。在有传感器噪声的情况下积分会产生漂移误差累积
- 使用加速度计：根据重力加速度方向可以计算得姿态，但是物体运动时会收到影响。
- 使用磁力计：容易受到周围磁性材料影响
-



使用IMU进行姿态估计

Pose Estimation using IMU

浙江大学 ZHEJIANG UNIVERSITY

	加速度传感器	电子罗盘（磁力计）	陀螺仪
功能	通过测量三个轴的加速度大小来判断人体运动	通过测量设备周围地磁场的强度和方向来判断朝向	通过测量三个轴的旋转速率来判断朝向
主要局限性	受重力干扰大，瞬时误差大	误差大， 容易受其他磁场和金属物体影响。主要用于校正其他设备	误差会累积，长时间读数的准确性差
应用	活动测量	导航	导航

因此我们需要传感器融合！

Core Motion 简介

Introduction to Core Motion

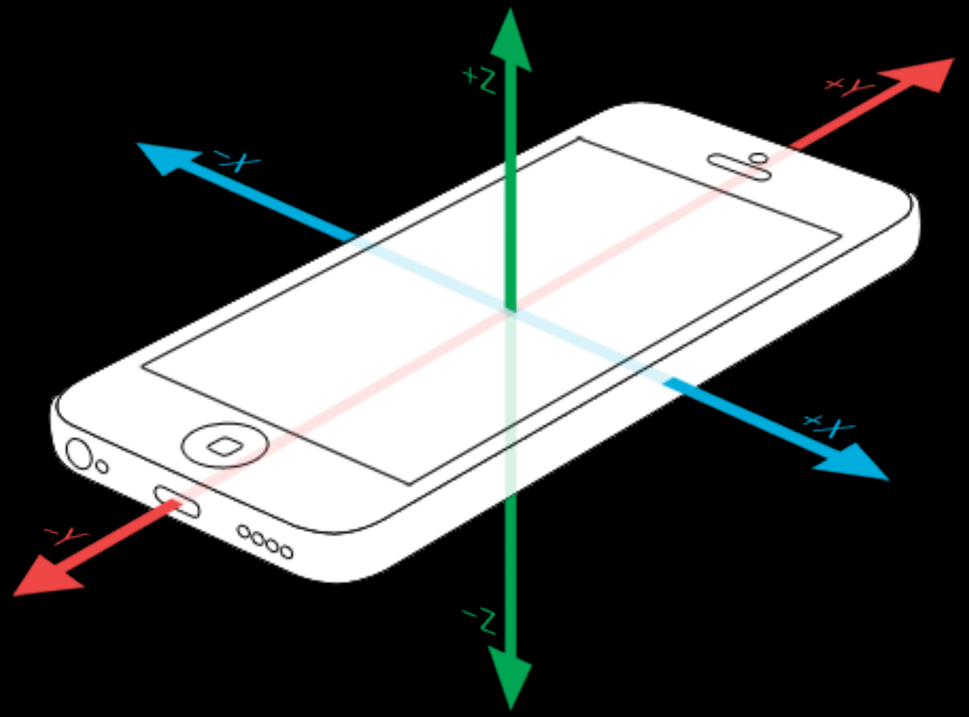
- CMDeviceMotion （处理后的数据）包含这些常用的功能:
  - attitude（类型:CMAttitude）：返回设备的方位信息，包含roll 、pitch、yaw三个欧拉角的值
  - rotationRate（类型:CMRotationRate）：经过滤波操作之后的陀螺仪数据
  - gravity（类型:CMAcceleration）：返回重力对设备在三个方向上的加速度，重力加速度矢量在当前设备的参考坐标系中的表达，也是处理后的数据
  - userAcceleration（类型:CMAcceleration）：返回用户对设备在三个方向上的加速度。不再需要滤波，但根据程序需求而加的滤波算法可以保留

Core Motion 简介

Introduction to Core Motion

思考题🤔：这是左手系还是右手系？ 右手

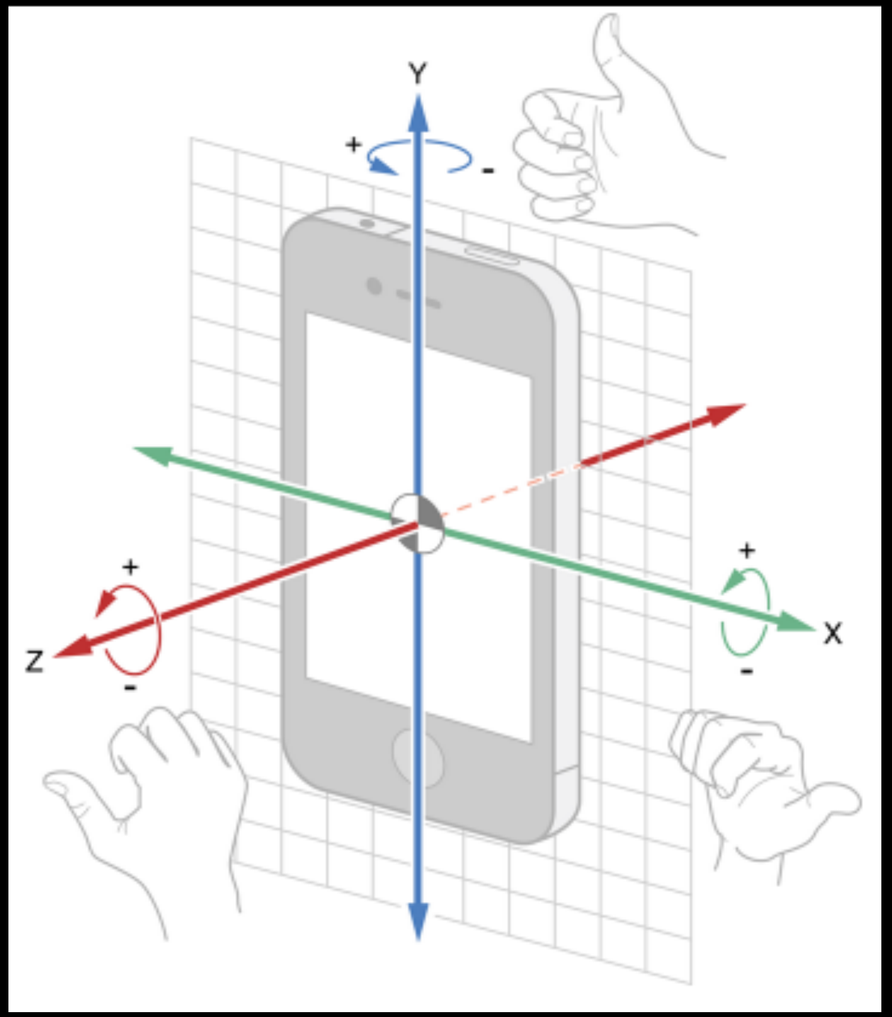
- 加速度计 (类型:CMAcceleration)
  - 加速度计可以检测三维空间中的加速度，坐标对应如右图
  - 坐标轴：设备屏幕为XY平面，垂直平面向外是Z正轴
  - 例如：当垂直手持手机且顶部向上，Y坐标上会受到 -1G的加速度



- 原始数据（未处理的数据流）可以通过 CMMotionManager.accelerometerData 获得

Core Motion 简介

Introduction to Core Motion



- 陀螺仪 (类型:CMRotationRate)
  - 陀螺仪用于检测设备绕XYZ轴转动的速度，坐标对应如右图
  - 旋转角度的正负可以使用右手定则判断
  - 原始数据（未处理的数据流）可以通过 CMMotionManager.gyroData 获得

Core Motion 简介

Introduction to Core Motion

- Core Motion 还提供了其他传感器或传感数据的对象：
  - CMPedometer: 系统生成的行走数据，包含步数统计等
  - CMAltimeter：海拔高度数据
  - CMMagnetometerData：磁力计数据

https://developer.apple.com/documentation/coremotion

Core Motion

Core Motion

- 使用流程：
  - 检测设备可用性，获得 CMMotionManager
  - 设置数据更新周期

```
@Published var manager: CMMotionManager = .init()

func detectMotion(){
     if !manager.isDeviceMotionActive{
         manager.deviceMotionUpdateInterval = 1/40
         }
     }
 }
```

Core Motion

Core Motion

- 更新数据：
  - 同步更新方法（Synchronous）

```
manager.startDeviceMotionUpdates(to: .main) {[weak self] motion, err in
    if let attitude = motion?.attitude{
        self?.xValue = attitude.roll
        print(attitude.roll)
    }

}
```

Core Motion

Core Motion

- 更新数据：
  - 异步更新方法（Asynchronous）

```
manager.startDeviceMotionUpdatesToQueue(
    NSOperationQueue.currentQueue(), withHandler: {
    (deviceMotion, error) -> Void in

    if(error == nil) {
        self.handleDeviceMotionUpdate(deviceMotion)
    } else {
        //handle the error
    }
})
```

Core Motion

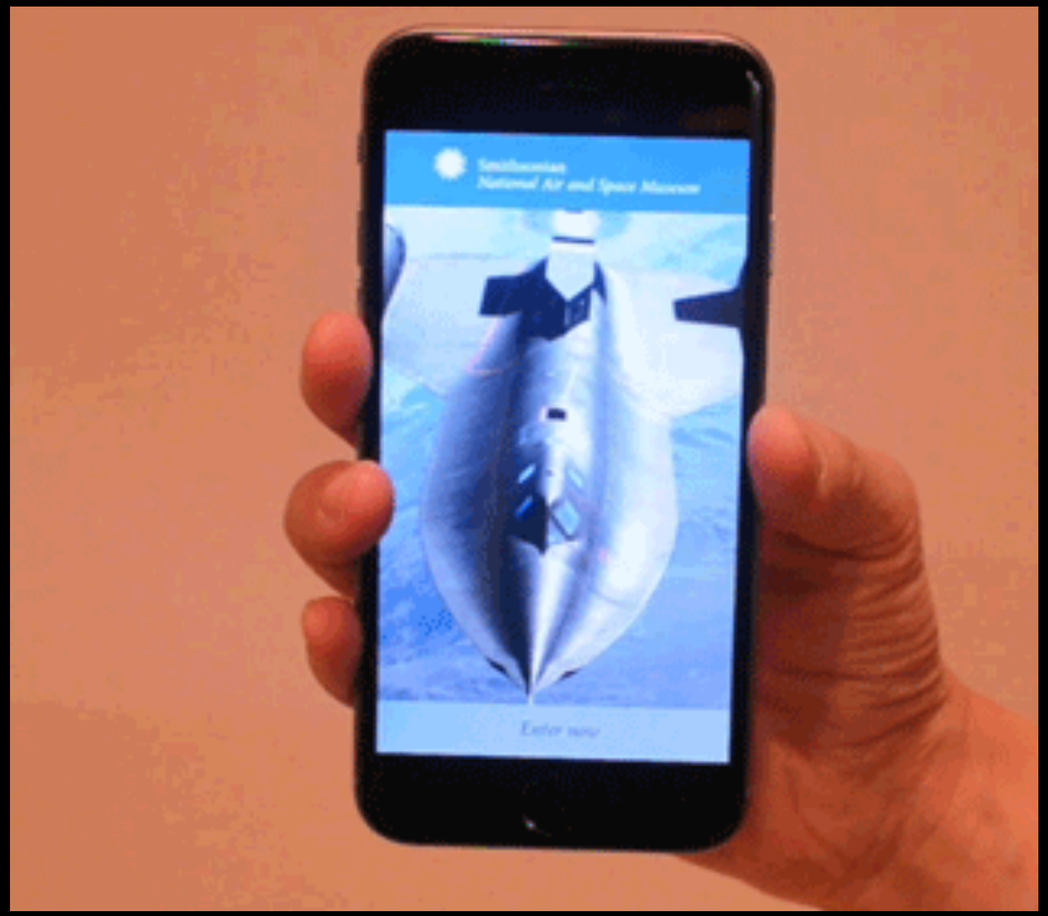
Core Motion

使用 CMDeviceMotionData 来获取处理后的各类数据：设备姿态（朝向）、滤波后的角速度、滤波后的重力方向

```
if manager.deviceMotionAvailable {
        manager.deviceMotionUpdateInterval = 0.01
        manager.startDeviceMotionUpdatesToQueue(
                NSOperationQueue.mainQueue()) {
            [weak self] (data: CMDeviceMotionData!,
                         error: NSError!) in
            let rotation = atan2(data.gravity.x,
                                 data.gravity.y) - M_PI
            self?.imageView.transform =
        CGAffineTransformMakeRotation(
            CGFloat(rotation))
        }
    }
```

Core Motion

Core Motion



- 使用deviceMotion类型来获取加速器和陀螺仪的融合数据
- 通过使用陀螺仪，Core Motion能依靠重力加速度来区分用户的动作

浙江大學
ZHEJIANG UNIVERSITY

IMU基础知识Core Motion

Core Location

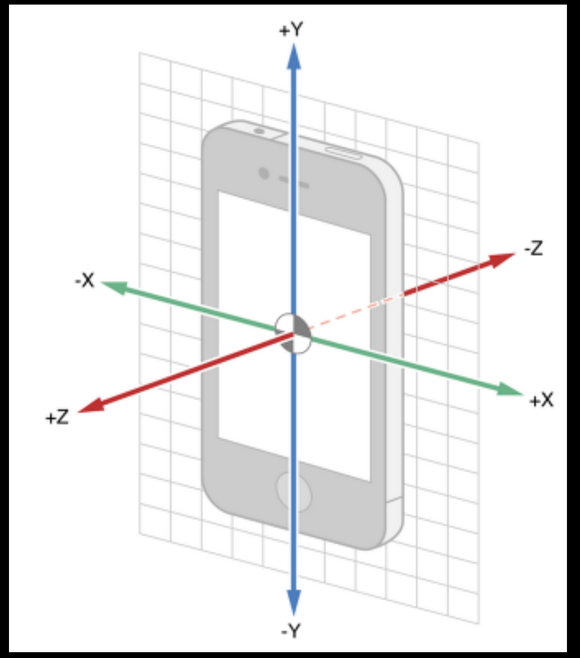
AVFoundation 与 Core Image

Core Location 简介

Introduction to Core Location

浙江大学 ZHEJIANG UNIVERSITY

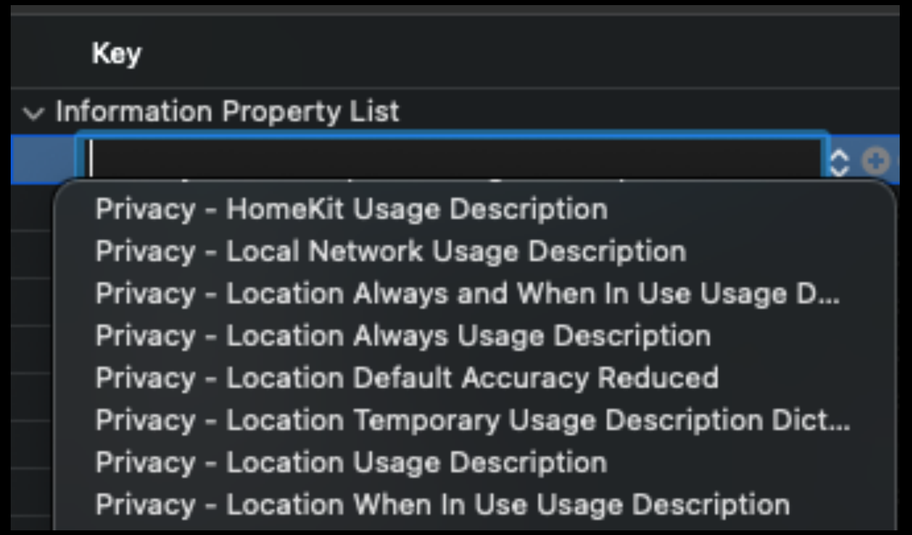
- 在iOS中，我们可以通过两套工具获取设备的姿态动作，对比前面提到的 Core Motion：
  - Core Motion 通过IMU信息来获取设备的位移、姿态。
  - Core Location 通过GPS、WiFi基站等方案对设备进行定位。 实现LBS应用必须使用Core Location，通过CLLocationManager，用户可以直接获得设备的经纬以及海拔。



Core Location 使用

Core Location Usage

- 在iOS8之后，Apple加强了对用户隐私的保护，需要调用iOS API主动请求用户授权
- 同时需要在plist中配置对应的键值，否则请求授权的方法不会生效

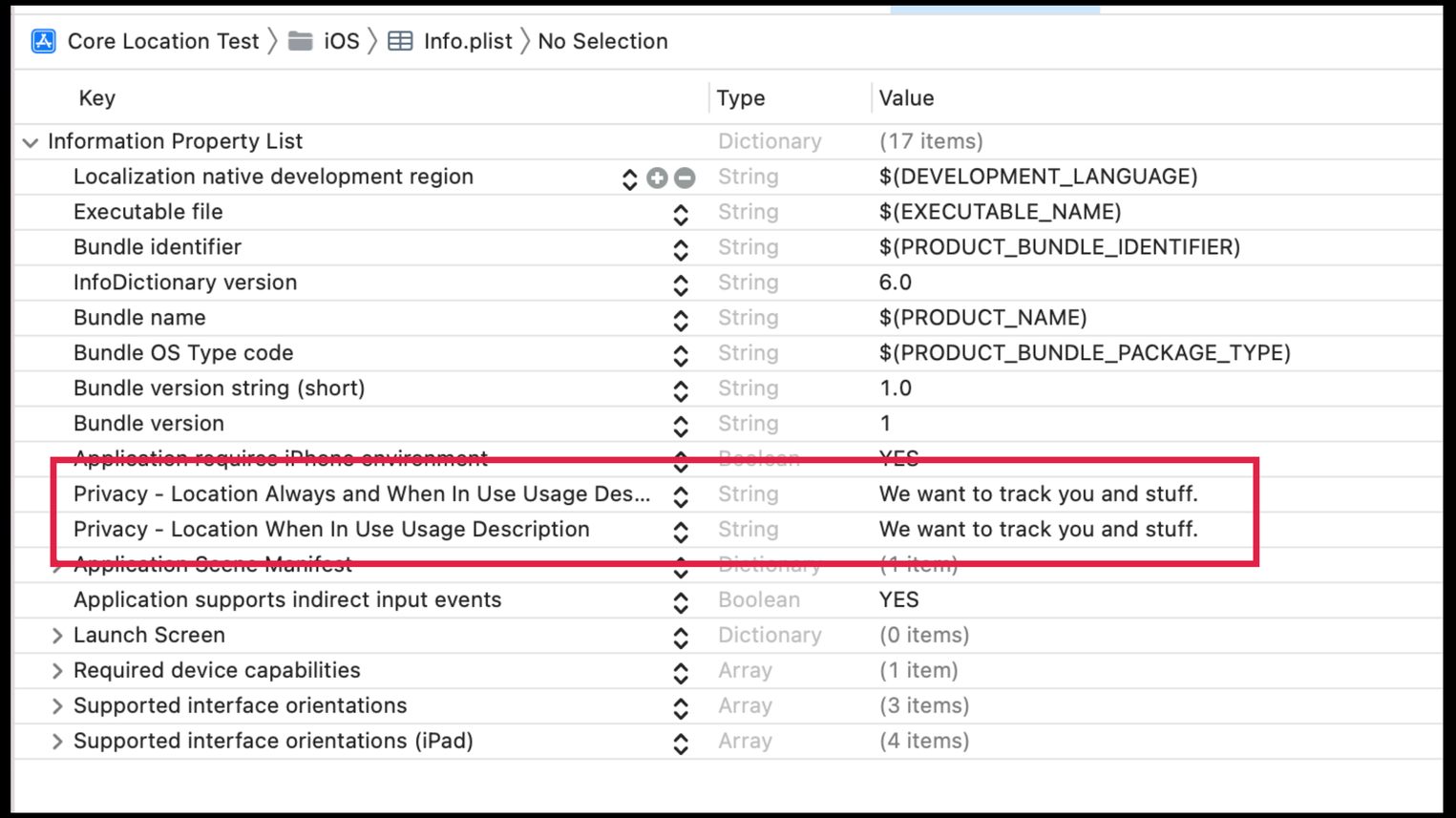


Core Location 使用

Core Location Usage

- 设置之后应该能看到

  这样的键值对



Core Location 使用

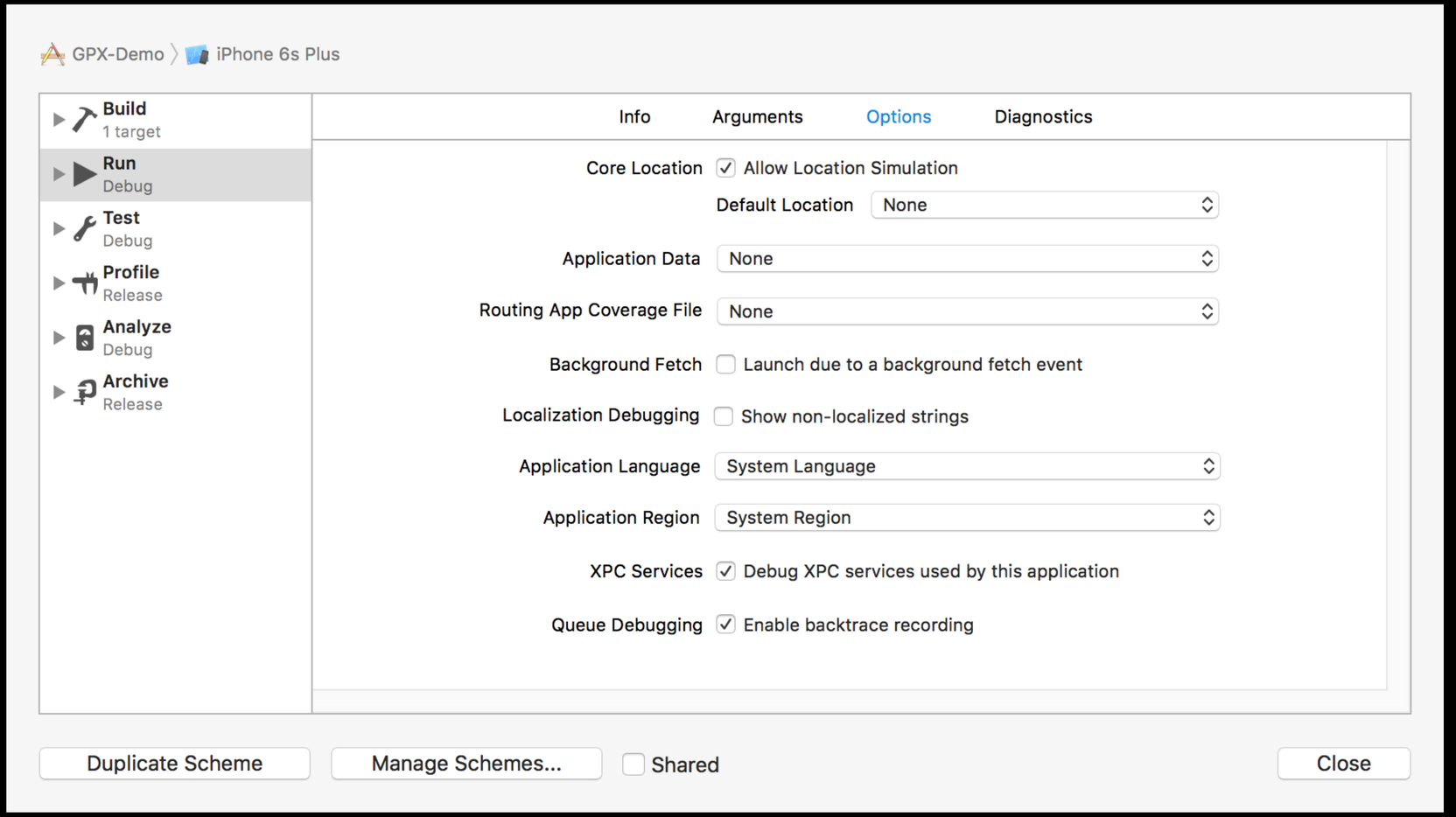
Core Location Usage

- 同时需要在plist中配置对应的键值，否则请求授权的方法不会生效
  - 选择 Privacy - Location... 相关的选项
- 为了方便苹果的审核，应用中涉及到隐私相关的权限都需要在plist中单独添加。例如，大部分MapKit相关应用需要使用网络，而使用网络也需要在plist中加入相应的属性来获得网络权限

Core Location 使用

Core Location Usage

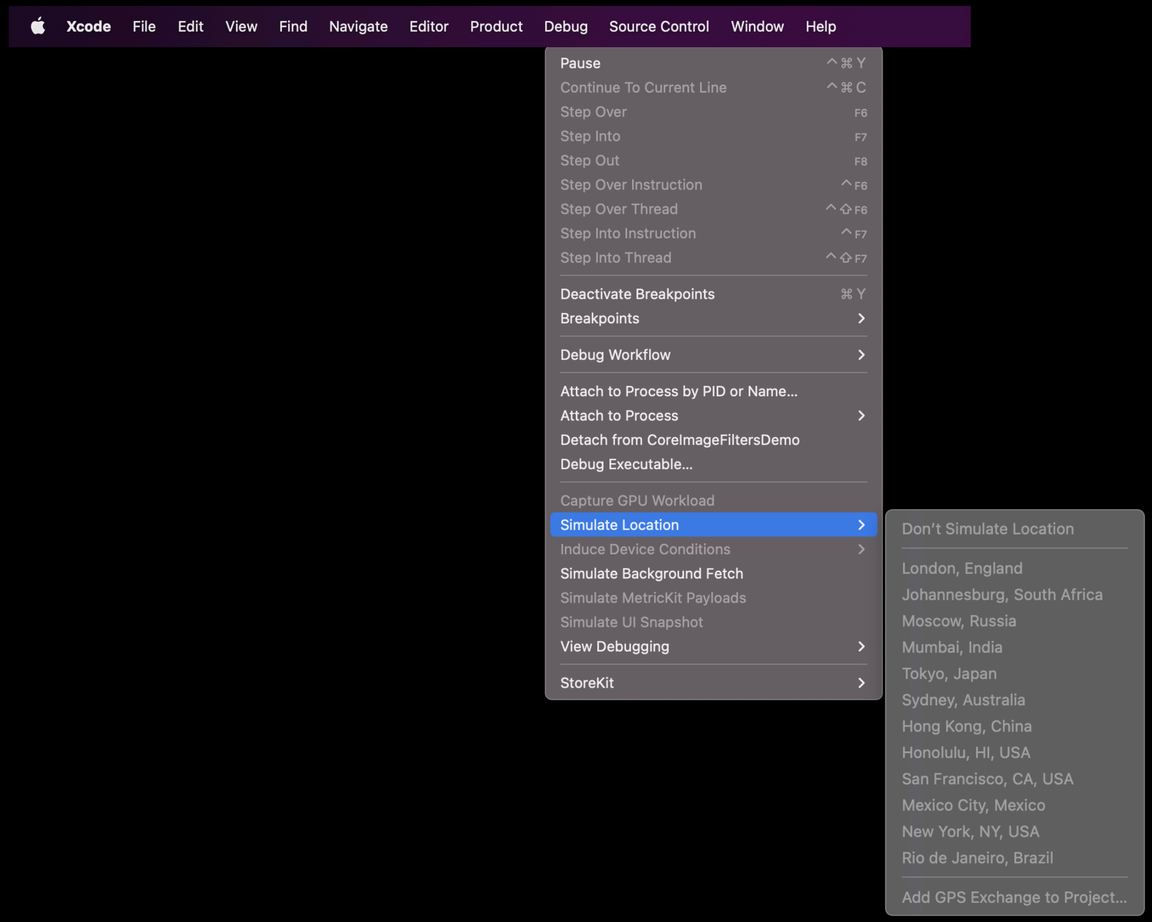
- 想要在模拟器中使用定位，需要在Product->Scheme- >Edit Scheme->Options中勾选 **Allow Location Simulation**



Core Location 使用

Core Location Usage

- Xcode > Debug > Simulate Location



Core Location Demo

Core Location Demo

- CLLocationCoordinate2D用来表示物体在地球上的位置，分为经度和纬度
- 通常CLLocationManager将该坐标通过delegate传回

```
rightText: String(coordinate?.latitude ?? 0)
rightText: String(coordinate?.longitude ?? 0)
```

Core Location Demo

Core Location Demo

```
switch locationViewModel.authorizationStatus {
        case .notDetermined:
            AnyView(RequestLocationView())
                .environmentObject(locationViewModel)
        case .restricted:
            ErrorView(errorText: "Location use is restricted.")
        case .denied:
            ErrorView(errorText: "The app does not have location permissions.
Please enable them in settings.")
        case .authorizedAlways, .authorizedWhenInUse:
            TrackingView()
                .environmentObject(locationViewModel)
        default:
            Text("Unexpected status")
        }
```

Core Location Demo

Core Location Demo

```
    func requestPermission() {
            locationManager.requestWhenInUseAuthorization()
        }


struct RequestLocationView: View {
    @EnvironmentObject var locationViewModel: LocationViewModel

    var body: some View {
        VStack {
            Image(systemName: "location.circle")
                .resizable()
                .frame(width: 100, height: 100, alignment: .center)
                .foregroundColor(/*@START_MENU_TOKEN@*/.blue/*@END_MENU_TOKEN@*/)
            Button(action: {
                locationViewModel.requestPermission()
            }, label: {
                Label("Allow tracking", systemImage: "location")
            })
            .clipShape(RoundedRectangle(cornerRadius: 8))
            Text("We need your permission to track you.")
                .font(.caption)
        }
    }
```

Core Location Demo

Core Location Demo

- 通常使用经度纬度+海拔来表示位置，其中经度纬度使用 CLLocationCoordinate2D，海拔使用 CLLocation.altitude
- 当海拔的值为负数时，表示水平面以下，坐标的值为负数时，表示返回的是无效坐标

```
var coordinate: CLLocationCoordinate2D? {
        locationViewModel.lastSeenLocation?.coordinate
    }

@Published var lastSeenLocation: CLLocation?

String(locationViewModel.lastSeenLocation?.altitude ?? 0)
```

Core Location Demo

Core Location Demo

```
PairView(
    leftText: "Latitude:",
    rightText: String(coordinate?.latitude ?? 0)
)
PairView(
    leftText: "Longitude:",
    rightText: String(coordinate?.longitude ?? 0)
)
PairView(
    leftText: "Altitude",
    rightText: String(locationViewModel.lastSeenLocation?.altitude ?? 0)
)
```

Core Location Demo

Core Location Demo

- 设备定位的方法有很多种:GPS、WiFi基站等等，但是 CLLocationManager根据设定的精度来更新坐标，即开发者和用户不必关心底层定位的方法，只需要关心精度就可以
- 需要注意的是：精度越高，耗电量越高

```
super.init()
        locationManager.delegate = self
        locationManager.desiredAccuracy = kCLLocationAccuracyBest
        locationManager.distanceFilter = 0.4
        locationManager.startUpdatingLocation()
```

Core Location Demo

Core Location Demo

- 与Core Location相关的其他量：
  - CLLocationSpeed：用户移动速度，单位为米/秒
  - CLLocationDirection：用户移动方向，相对正北的角度
  - ...

```
var speed:CLLocationSpeed
/*
    meters/second
    instantaneous speed
    negative value means "speed is invalid"
 */

var course:CLLocationDirection
/*
    degree, 0 -north, clockwise
    negative value means "course is invalid"
 */

var timeStamp:NSDate
```

Core Location Demo

Core Location Demo

- 使用时候，需要初始化一个CLLocationManager，设置好委托、精度要求、最小的更新距离，然后开始更新定位

```
if AppLocationData.llocationManager == nil {
    AppLocationData.llocationManager = CLLocationManager.init()
    AppLocationData.llocationManager?.delegate = self

    AppLocationData.llocationManager?.desiredAccuracy = kCLLocationAccuracyBest

    AppLocationData.llocationManager?.distanceFilter = 1000.0

    AppLocationData.llocationManager?.startUpdatingLocation()

}
```

Core Location Demo

Core Location Demo

- 在委托中需要实现两个方法：
  - 更新失败时的调用
  - 更新成功时的调用

```
func locationManager(_ manager: CLLocationManager, didFailWithError error: Error) {
    print("Locate Failed:\(error)")
}

var currentLocation:CLLocation?
var currentCity:String?
var currentAddress:String?
var currentCountry:String?
var currentProvince:String?
var currentArea:String?

func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
    print("Locate Success.")
```

Core Location Demo

Core Location Demo

- 当定位更新成功时，我们会得到一系列定位，在locations里
- 使用CLGeocoder可以查到该定位的地址编码

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
    print("Locate Success.")

    self.currentLocation = locations.last

    let geoCoder:CLGeocoder = CLGeocoder.init()
    geoCoder.reverseGeocodeLocation(locations.last!, completionHandler: {
        (placemarks,err) in

        if placemarks == nil {
            return
        }

        let placeMark:CLPlacemark = placemarks![0]

        self.currentCity = placeMark.locality?.replacingOccurrences(of: "市", with: "")
        self.currentAddress = placeMark.addressDictionary?["FormattedAddressLines"] as? String
        self.currentCountry = placeMark.addressDictionary?["Country"] as? String
        self.currentProvince = placeMark.addressDictionary?["State"] as? String
        self.currentArea = placeMark.addressDictionary?["SubLocality"] as? String
    })

}
```

Core Location Demo

Core Location Demo

- 通过location的属性course，我们可以获取为当前行进的角度
- 通过distance，我们可以获取两个CLLocation之间的距离
- 其他属性请同学们使用前查阅官方文档

```
switch Int((self.currentLocation?.course)!/90) {
case 0:
    print("北偏东")
case 1:
    print("东偏南")
case 2:
    print("南偏西")
case 3:
    print("西偏北")
default:
    print("上月球了？？")
}

locations.last?.distance(from: self.currentLocation!)
```

Core Location 总结

Core Location Summary

- 在plist中设置隐私权限
- 编写代码获取隐私权限
- 设定精度并定义委托方法，即可获得设备传出的各种属性（坐标、海拔、运动方向等）

浙江大学
ZHEJIANG UNIVERSITY

IMU基础知识Core Motion

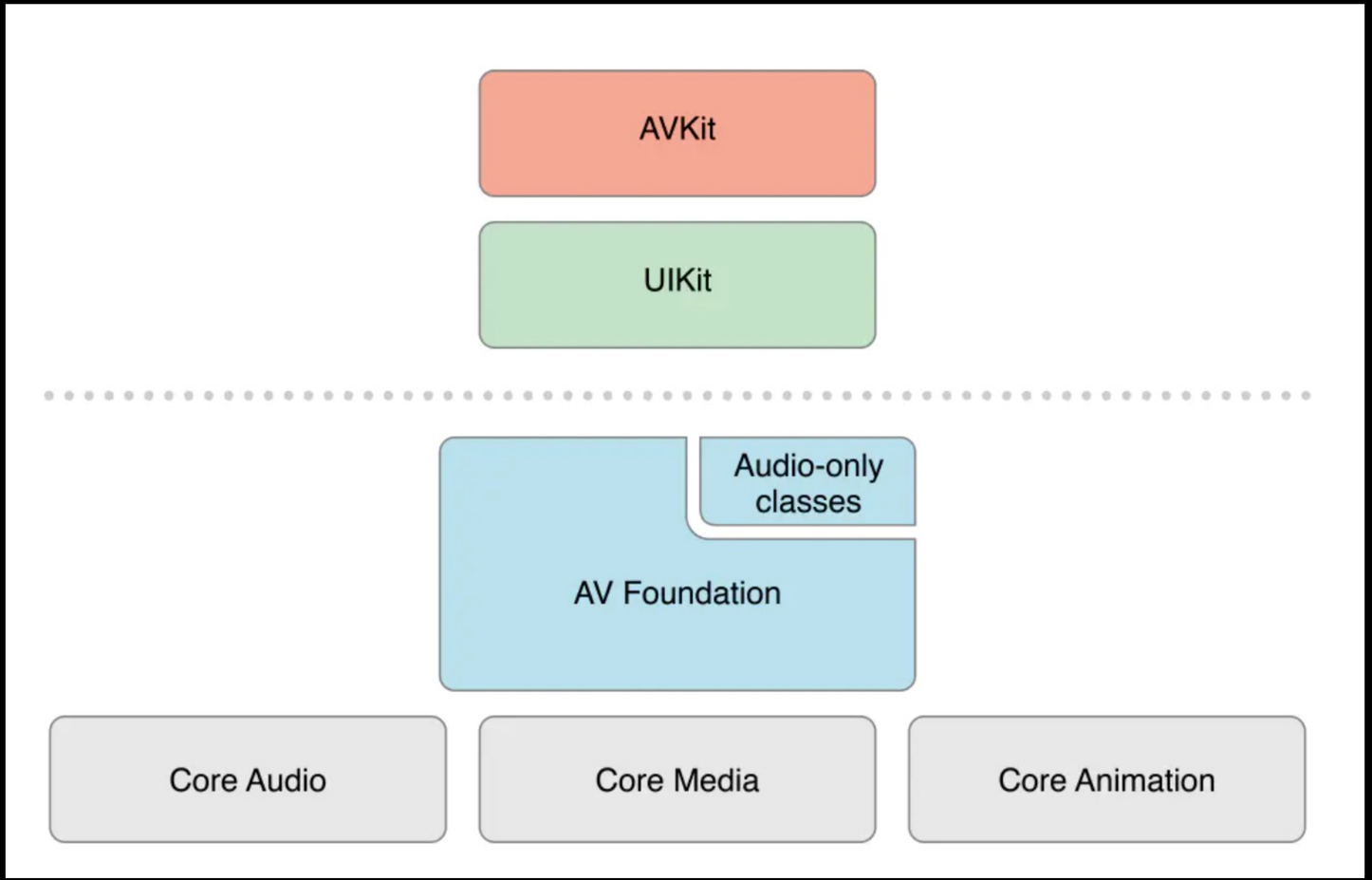
Core Location

AVFoundation 与 Core Image

AVFoundation - 基本概念

AVFoundation - Basic Concepts

- AVFoundation 结合了音视频的主要技术领域
- 它提供了一个操作视听媒体的接口，是更低级别的多媒体框架。
- iOS 系统中的 AVFoundation 媒体架构如右图

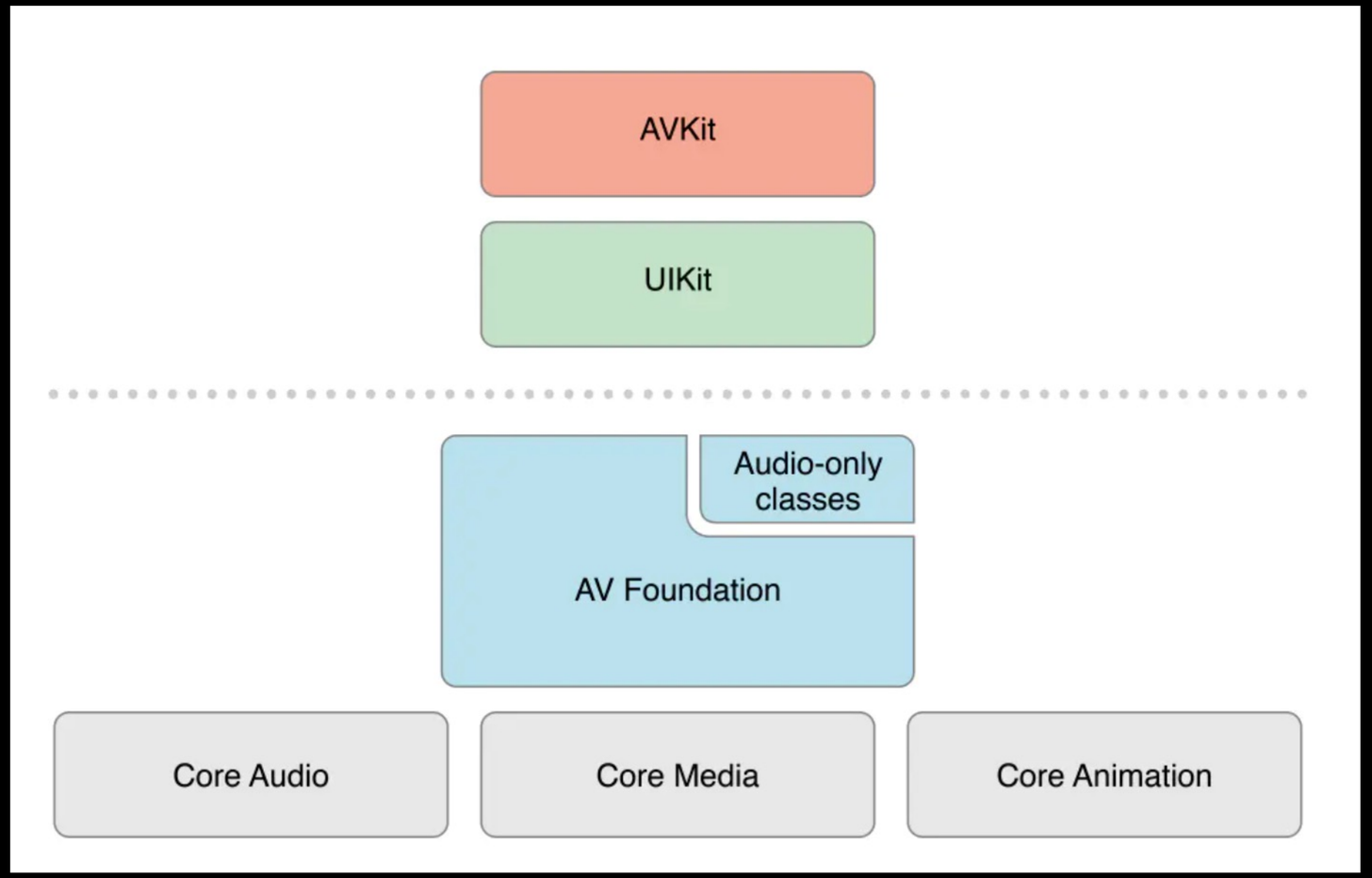


https://developer.apple.com/documentation/avfoundation

AVFoundation - 主要功能

AVFoundation - Major Function

- 媒体资源的表示与读取
- 媒体录制与回放
- 媒体编辑与处理
  - 数字媒体压缩（编解码）
- ...



https://developer.apple.com/documentation/avfoundation

AVFoundation - 重要的类

AVFoundation - Important Classes

AVAsset

AVMetadataItem

AVPlayer

AVPlayerItem

...

AVFoundation - 重要的类

AVFoundation - Important Classes

- AVAsset
  - AVFoundation框架中使用AVAsset类来表示一个媒体资源。
  - AVAsset是一个抽象类，可以使用它的子类来从URL创建一个asset对象，或者根据已有的媒体资源创造出一个新的媒体资源
  - Asset中每个独立的媒体数据都有一个统一的类型：track. 一个asset中非常典型的情况是，一个track代表音频,另一个track则代表视频。也可以有多个同时包含音视频的track。Assets同时也可能包含元数据

AVFoundation - 重要的类

AVFoundation - Important Classes

- AVAsset
  - 资源的加载是同步的，如果在主线程中加载较大的资源那么可能会导致线程阻塞。可以使用异步加载，可以使用statusOfValue(forKey: error:)方法来查看加载的进度。

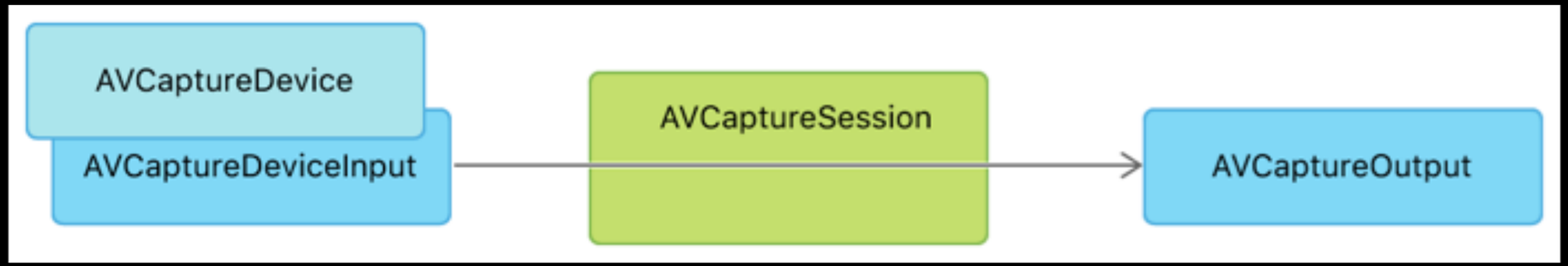
```swift
// URL of a bundle asset called 'example.mp4'
let url = Bundle.main.url(forResource: "example", withExtension: "mp4")!
let asset = AVAsset(url: url)
let playableKey = "playable"

// Load the "playable" property
asset.loadValuesAsynchronously(forKeys: [playableKey]) {
    var error: NSError? = nil
    let status = asset.statusOfValue(forKey: playableKey, error: &error)
    switch status {
    case .loaded:
        // Sucessfully loaded. Continue processing.
    case .failed:
        // Handle error
    case .cancelled:
        // Terminate processing
    default:
        // Handle all other cases
    }
}
```

AVFoundation - 媒体获取

AVFoundation - Media Acquisition

- **AVCaptureSession**
  - 负责采集音视频，可以将一个或多个输入连接到一个或多个输出。媒体的input，包括iOS设备或Mac内置的摄像头和麦克风等采集设备。output则是后续程序要用到的有用数据，例如写入磁盘的音视频文件、可用于实时处理的原始像素缓冲区。



AVFoundation - 音频播放

AVFoundation - Audio Playback

- AVAudioSession
  - 在播放音频之前，系统会设置一个默认的audio session，它是操作系统和App进行音频管理的媒介

```
let audioSession = AVAudioSession.sharedInstance()
do {
    try audioSession.setCategory(AVAudioSessionCategoryPlayback)
} catch {
    print("Setting category to AVAudioSessionCategoryPlayback failed.")
}
```

AVFoundation - 视频播放

AVFoundation - Video Playback

- AVPlayer
  - 负责播放音视频及时间管理，AVPlayer一次只能播放一个资源，如果需要顺序播放多个资源，可以使用它的子类AVQueuePlayer来管理播放队列
- AVPlayerItem
  - 负责和AVPlayer交互，管理媒体资源的动态部分，管理资源的时间和状态

Core Image - 基本概念

Core Image - Basic Concepts

- Core Image是iOS中的图像处理框架，使用上比较简单方便，常用于照片的滤镜处理或面部检测等用途
  - CIImage：代表图像的类
  - CIFilter：滤镜类，能够通过key-value来设置输入值，用于为CIImage添加滤镜
  - CIContext：用于渲染CIImage
  - CIDetector：用于分析CIImage，得到CIFeature。
  - CIFeature\CIFaceFeature：主要用于面部检测等应用
  - ...

https://developer.apple.com/documentation/coreimage

Core Image Demo

Core Image Demo

- CoreImage处理图像的流程
  - 获取UIImage
  - 初始化CIContext和CIImage

```
@Published var image = UIImage()
var originalImage: UIImage

init() {
    originalImage = UIImage(named: "sample_image") ?? UIImage()
    image = originalImage
}
```

Core Image Demo

Core Image Demo

- CoreImage处理图像的流程
  - 获取UIImage
  - 初始化CIContext和CIImage

```
@Published var image = UIImage()
var originalImage: UIImage
let context = CIContext()
var ciImage: CIImage? {
    guard let cgImage = originalImage.cgImage else { return nil }
    return CIImage(cgImage: cgImage)
}
```

Core Image Demo

Core Image Demo

- CoreImage处理图像的流程
  - 设置滤镜
  - 一个滤镜就是一个函数

```
func applyCrystallize() {
        guard let ciImage else { return }
        let filter = CIFilter(name: "CICrystallize")
        filter?.setValue(ciImage, forKey:
kCIInputImageKey)
        filter?.setValue(35, forKey: kCIInputRadiusKey)
        filter?.setValue(CIVector(x: 200, y: 200), forKey:
kCIInputCenterKey)
        getImage(usingFilter: filter)
    }
```

Core Image Demo

Core Image Demo

- CoreImage处理图像的流程
  - 设置滤镜
  - 滤镜可以按处理顺序链接成一个函数调用的“链”，也可以写在同一个函数内

```
func chainFilters() {
        guard let ciImage else { return }

        let sepiaFilter = CIFilter(name: "CISepiaTone")
        sepiaFilter?.setValue(ciImage, forKey: kCIInputImageKey)
        sepiaFilter?.setValue(0.9, forKey: kCIInputIntensityKey)

        guard let sepiaOutput = sepiaFilter?.outputImage else { return }

        let blurFilter = CIFilter(name: "CIMotionBlur")
        blurFilter?.setValue(sepiaOutput, forKey: kCIInputImageKey)
        blurFilter?.setValue(30.0, forKey: kCIInputRadiusKey)
        blurFilter?.setValue(20.0, forKey: kCIInputAngleKey)

        getImage(usingFilter: blurFilter)
    }
```

Core Image Demo

Core Image Demo

- CoreImage处理图像的流程
  - 通过outputImage输出图像
  - 需要将图像转回 UIImage

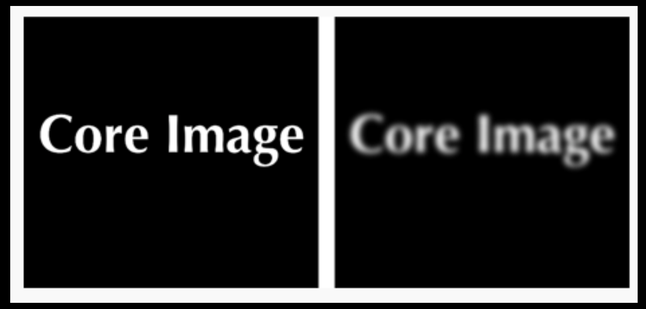
```
fileprivate func getImage(usingFilter filter: CIFilter?) {
        guard let output = filter?.outputImage else { return }
        guard let cgImage = context.createCGImage(output, from: output.extent) else { return }
        image = UIImage(cgImage: cgImage)
    }
```

Core Image Demo - 一些滤镜效果

Core Image Demo - Some Filter Effects

- CIGaussianBlur

高斯模糊

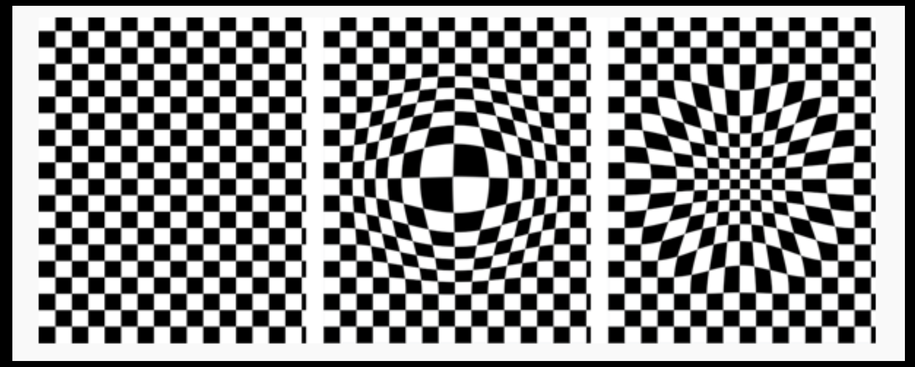


- CIColorMap

自定义颜色映射

- CIBumpDistortion

自定义凹凸变形



Core Image内置了几十种Image Filter，使用前可以参考官方指南

https://developer.apple.com/documentation/coreimage

Core Image Demo

Core Image Demo

DEMO - Live Filter

浙江大學
ZHEJIANG UNIVERSITY

QUESTIONS?