作者 YotrolZ (/users/244aa1f48d1c) 2015.08.21 17:16*
写了18461字，被430人关注，获得了281个喜欢
(/users/244aa1f48d1c)

iOS中传感器的基本使用

iOS中常见的传感器

传感器类型	作用
环境光传感器	感应周边环境光线的强弱（自动调节屏幕亮度）
距离传感器	感应是否有其他物体靠近设备屏幕（打电话自动锁屏）
磁力计传感器	感应周边的磁场（合盖锁屏）
内部温度传感器	感应设备内部的温度（提醒用户降温，防止损伤设备）
湿度传感器	感应设备是否进水（方便维修人员）
陀螺仪	感应设备的持握方式（赛车类游戏）
加速计	感应设备的运动（摇一摇、计步器）

iOS中常见的传感器

一.距离传感器

- 监听方式:添加 观察者 ,监听 通知
- 通知名称: `UIDeviceProximityStateDidChangeNotification`
- 监听状态:观察者的对应回调方法中,判断 `[UIDevice currentDevice].proximityState`
  - 返回 `NO` : 有物品靠近了;
  - 返回 `YES` : 有物品远离了
- 注意:使用前要打开当前设备距离传感器的开关(默认为:NO):

```
[UIDevice currentDevice].proximityMonitoringEnabled = YES;
```

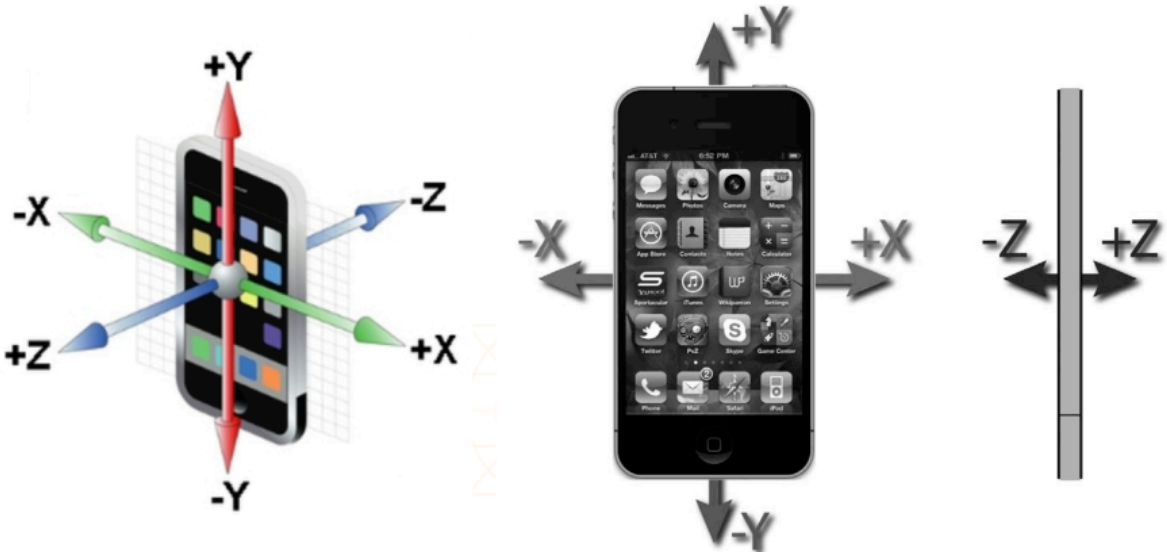
- 示例程序:

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // [UIApplication sharedApplication].proximitySensingEnabled = YES;
    [UIDevice currentDevice].proximityMonitoringEnabled = YES;

    // 监听有物品靠近还是离开
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(proxim
}

- (void)proximityStateDidChange
{
    if ([UIDevice currentDevice].proximityState) {
        NSLog(@"有物品靠近");
    } else {
        NSLog(@"有物品离开");
    }
}
```

二.加速计(UIAccelerometer)

- 概述:

  检测设备在 X/Y/Z轴 上的 受力 情况



加速计

- 监听方式:设置 代理
- 使用步骤:(iOS5之前)
  - 获取加速计 单例 对象:

```
UIAccelerometer *accelerometer = [UIAccelerometer sharedAccelerometer];
```

  - 设置加速计 代理 对象

```
accelerometer.delegate = self;
```

  - 设置 采样间隔 ： updateInterval

```
accelerometer.updateInterval = 0.3;
```

  - 实现代理相关方法,监听加速计的数据

```
- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAccele
```

    - UIAcceleration 参数:

```
@interface UIAcceleration : NSObject {
@private
NSTimeInterval timestamp;
UIAccelerationValue x, y, z;
}
```

- 示例程序:

```
#import "ViewController.h"

@interface ViewController () <UIAccelerometerDelegate>

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    // 1.获取单例对象
    UIAccelerometer *accelerometer = [UIAccelerometer sharedAccelerometer];

    // 2.设置代理
    accelerometer.delegate = self;

    // 3.设置采样间隔
    accelerometer.updateInterval = 0.3;
}

- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration
{
    NSLog(@"x:%f y:%f z:%f", acceleration.x, acceleration.y, acceleration.z);
}

@end
```

- 备注: `UIAcceleration` 和 `UIAccelerometer` 在iOS 5.0中已经被弃用。由 `Core Motion framework` 取代。
    - 官方提示信息:

```
UIAcceleration and UIAccelerometer are deprecated as of iOS 5.0. These classes
```

三 . Core Motion

- Core Motion获取数据的两种方式:
    - `push` : 实时采集所有数据,采集 频率高 ;
    - `pull` : 在有需要的时候,才去采集数据;

Core Motion的使用步骤--- `push`

- 1.创建运动管理对象

```
CMMotionManager*mgr = [[CMMotionManageralloc]init];
```

- 2.判断加速器是否可用(最好判断)

```
if(mgr.isAccelerometerAvailable){
  //加速计可用
}
```

- 3.设置采样间隔

```
mgr.accelerometerUpdateInterval= 1.0/30.0;// 1秒钟采样30次
```

- 4.开始采样(采样到数据就会调用handler，handler会在queue中执行)

```
-(void)startAccelerometerUpdatesToQueue:(NSOperationQueue*)queue withHandler:(CMA
```

- 示例程序:

```
#import "ViewController.h"

@interface ViewController () <UIAccelerometerDelegate>

/** 运动管理者 */
@property (nonatomic, strong) CMMotionManager *mgr; // 保证不死

@end

@implementation ViewController

#pragma mark - 懒加载
- (CMMotionManager *)mgr
{
    if (_mgr == nil) {
        _mgr = [[CMMotionManager alloc] init];
    }
    return _mgr;
}
- (void)viewDidLoad {
    [super viewDidLoad];

    // 1.判断加速计是否可用
    if (!self.mgr.isAccelerometerAvailable) {
        NSLog(@"加速计不可用");
        return;
    }

    // 2.设置采样间隔
    self.mgr.accelerometerUpdateInterval = 0.3;

    // 3.开始采样
    [self.mgr startAccelerometerUpdatesToQueue:[NSOperationQueue mainQueue] withHand
        if (error) return;

        // 4.获取加速计信息
        CMAcceleration acceleration = accelerometerData.acceleration;
        NSLog(@"x:%f y:%f z:%f", acceleration.x, acceleration.y, acceleration.z);
    }];
}
@end
```

简
(/)

(/collections)

(/apps)

Core Motion的使用步骤--- pull

- 说明： pull 是在需要时获取数据,我们此时以点击了屏幕就获取一次数据为例说明;

- 1.创建运动管理对象

```
CMMotionManager*mgr = [[CMMotionManageralloc]init];
```

- 2.判断加速器是否可用(最好判断)

```
if(mgr.isAccelerometerAvailable){
  //加速计可用
}
```

- 3.开始采样

```
-(void)startAccelerometerUpdates;
```

- 4.在需要时获取数据

```
CMAcceleration acc = mgr.accelerometerData.acceleration;
NSLog(@"%f,%f, %f", acc.x,acc.y,acc.z);
```

- 示例程序:

```
#import "ViewController.h"

@interface ViewController () <UIAccelerometerDelegate>

/** 运动管理者 */
@property (nonatomic, strong) CMMotionManager *mgr; // 保证不死

@end

@implementation ViewController

#pragma mark - 懒加载
- (CMMotionManager *)mgr
{
    if (_mgr == nil) {
        _mgr = [[CMMotionManager alloc] init];
    }
    return _mgr;
}
- (void)viewDidLoad {
    [super viewDidLoad];

    // 1.判断加速计是否可用
    if (!self.mgr.isAccelerometerAvailable) {
        NSLog(@"加速计不可用");
        return;
    }

    // 2.开始采样
    [self.mgr startAccelerometerUpdates];
}
@end

    // 3.数据采样(以点击了屏幕为例说明)
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    // 获取加速计信息
    CMAcceleration acceleration = self.mgr.accelerometerData.acceleration;
    NSLog(@"x:%f y:%f z:%f", acceleration.x, acceleration.y, acceleration.z);
}
```

四.磁力计/陀螺仪的使用和上述加速计的使用步骤类似

不同点:

- 1.判断传感器 是否可用：

- 加速计:

```
@property(readonly, nonatomic, getter=isAccelerometerAvailable) BOOL acceleror
```

- 陀螺仪:

```
@property(readonly, nonatomic, getter=isGyroAvailable) BOOL gyroAvailable;
```

- 磁力计:

```
@property(readonly, nonatomic, getter=isMagnetometerAvailable) BOOL magnetome
```

- 2.设置传感器的 采样间隔 :
  - 1.加速计:

```
@property(assign, nonatomic) NSTimeInterval accelerometerUpdateInterval;
```

  - 2.陀螺仪:

```
@property(assign, nonatomic) NSTimeInterval gyroUpdateInterval;
```

  - 3.磁力计:

```
@property(assign, nonatomic) NSTimeInterval magnetometerUpdateInterval
```

- 3.1. 开始采样 的方法-- push :
  - 1.加速计:

```
- (void)startAccelerometerUpdatesToQueue:(NSOperationQueue *)queue withHandle
```

  - 2.陀螺仪:

```
- (void)startGyroUpdatesToQueue:(NSOperationQueue *)queue withHandler:(CMGyroH
```

- 3.磁力计:

```
- (void)startMagnetometerUpdatesToQueue:(NSOperationQueue *)queue withHandler
```

- 3.2.开发采样的方法-- pull
  - 1.加速计:

```
- (void)startAccelerometerUpdates;
```

  - 2.陀螺仪:

```
- (void)startGyroUpdates;
```

  - 3.磁力计:

```
- (void)startMagnetometerUpdates;
```

- 4.1获取采样数据-- push
  - 在对应的传感器的 开始采样 方法中的 handler 中;
- 4.2.获取采样数据-- pull
  - 在需要获取的数据地方调用下面的方法:
  - 加速计:

```
CMAcceleration acceleration = self.mgr.accelerometerData.acceleration;
NSLog(@"x:%f y:%f z:%f", acceleration.x, acceleration.y, acceleration.z);
```

  - 陀螺仪:

```
CMRotationRate rate = self.mgr.gyroData.rotationRate;
NSLog(@"x:%f y:%f z:%f", rate.x, rate.y, rate.z);
```

五.没事你就,摇一摇

```
- (void)motionBegan:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
    NSLog(@"开始摇一摇");
}

- (void)motionCancelled:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
    NSLog(@"摇一摇被取消");
}

- (void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event
{
    NSLog(@"摇一摇停止");
}
```

六. 没事走两步(计步器)

- 1.判断计步器是否可用

```
  if (![CMPedometer isStepCountingAvailable]) {
      NSLog(@"计步器不可用");
      return;
  }
```

- 2.创建计步器对象

```
CMPedometer *stepCounter = [[CMPedometer alloc] init];
```

- 3.开始记步,并获取采样数据

```
  [stepCounter startPedometerUpdatesFromDate:[NSDate date] withHandler:^(CMPedome
      if (error) return;
      // 4.获取采样数据
      NSLog(@"steps = %@", pedometerData.numberOfSteps);
  }];
```

七.蓝牙

- 简述:

  iOS中提供了4个框架用于实现蓝牙连接:

  - 1. GameKit.framework
    - 只能用于 iOS设备之间 的连接，多用于游戏（比如五子棋对战）,可以在游戏中增加 对等连接 ， 又称 对端连接 或 点对点连接 Peer To Peer ,从iOS7开始过期
  - 2. MultipeerConnectivity.framework
    - 只能用于 iOS设备之间 的连接，从iOS7开始引入
  - 3. ExternalAccessory.framework
    - 可用于 第三方蓝牙设备 交互，但是蓝牙设备必须经过 苹果MFi认证 （国内较少）
  - 4. CoreBluetooth.framework （时下热门）
    - 可用于 第三方蓝牙设备 交互，必须要支持蓝牙4.0;
    - 硬件至少是4s，系统至少是iOS6;
    - 蓝牙4.0以 低功耗 著称，一般也叫 BLE （BluetoothLowEnergy）
    - 目前应用比较多的案例：运动手环、嵌入式设备、智能家居

♡ 喜欢 | 19

分享到微博 分享到微信
更多分享

2条评论 （ 按时间正序 · 按时间倒序 · 按喜欢排序 ） 添加新评论 (/sign_in)

FloatM3 (/users/28890fe286de)
2楼 · 2016.05.18 14:28 (/p/233be81b8ead/comments/2435478#comment-2435478)
(/users/28890fe286de)
请问在获取传感器数据同时获得当前的时间该怎么实现呢。

♡ 喜欢(0) 回复

为什么我不饿 (/users/58fa58c1bcad)
3楼 · 2016.06.15 17:31 (/p/233be81b8ead/comments/2766860#comment-2766860)
(/users/58fa58c1bcad)
距离传感器 一接近就自动黑屏，怎么设置不让黑屏知道 吗

♡ 喜欢(0) 回复

登录后发表评论 (/sign_in)

被以下专题收入，发现更多相似内容：

iOS (/collection/27e6fb9b84f7)
学习
(/collection/27e6fb9b84f7)
473篇文章 (/collection/27e6fb9b84f7) · 1389人关注
+ 添加关注 (/sign_in)

iOS开发记录 (/collection/f9fe0b264e93)
记录iOS开发相关文章和开发技巧。
(/collection/f9fe0b264e93)
1421篇文章 (/collection/f9fe0b264e93) · 1144人关注
+ 添加关注 (/sign_in)

iOS开发好文 (/collection/5f98164232a9)
iOS开发文章收集，精选优质开发文章，总结常用知识点，让开发更简单
(/collection/5f98164232a9)
385篇文章 (/collection/5f98164232a9) · 399人关注
+ 添加关注 (/sign_in)