Alamofire

<https://github.com/Alamofire/Alamofire>

# Core Motion (optional)

We can do a lot of exciting things with Core Motion. Our devices can keep track of what angle we are holding the phone, how fast we are travelling, how many steps we have taken and so on. Below are a few basic examples of how we can use the Core Motion library.

To learn more check out the Apple documentation on [Core Motion](https://developer.apple.com/documentation/coremotion).

**Note**: Don't forget to add privacy settings for Info.plist!

### Device Motion

Here is some starter code for accessing the motion of your device.

import UIKit

import CoreMotion

class ViewController: UIViewController {

var motionManager = CMMotionManager()

let opQueue = OperationQueue()

override func viewDidLoad() {

super.viewDidLoad()

if motionManager.isDeviceMotionAvailable {

print("We can detect device motion")

startReadingMotionData()

}

else {

print("We cannot detect device motion")

}

}

func startReadingMotionData() {

// set read speed

motionManager.deviceMotionUpdateInterval = 1

// start reading

motionManager.startDeviceMotionUpdates(to: opQueue) {

(data: CMDeviceMotion?, error: Error?) in

if let mydata = data {

print("mydata.gravity", mydata.gravity)

print("pitch raw", mydata.attitude.pitch)

print("pitch", self.degrees(mydata.attitude.pitch))

}

}

}

func degrees(\_ radians: Double) -> Double {

return 180/Double.pi \* radians

}

}

### Step Counter

Here is some code for the pedometer. This application will count the steps of the user using the application.

import UIKit

import CoreMotion

class ViewController: UIViewController {

let pedometer: CMPedometer = CMPedometer()

@IBOutlet weak var stepCount: UILabel!

override func viewDidLoad() {

super.viewDidLoad()

}

@IBAction func resetButtonPressed(sender: UIButton) {

pedometer.stopUpdates()

stepCount.text = "0"

}

@IBAction func startButtonPressed(sender: UIButton) {

pedometer.startUpdates(from:Date(), withHandler: { data, error in

print(data!)

let numSteps = data?.numberOfSteps

DispatchQueue.main.async {

self.stepCount.text = "\(numSteps ?? 0)"

}

})

}

}

# AVFoundation

AVFoundation is a Framework that we can use to make applications that use Audio or Video. Here is the starter code that will turn on our device's camera once the application is launched. Have fun!

**Note**: Don't forget to add privacy settings for Info.plist!

import UIKit

import AVFoundation

class ViewController: UIViewController {

// can receive data from input devices such as videoInput

var captureSession: AVCaptureSession?

// used to display a preview of the data that is coming in from captureSession's input device

var previewLayer: AVCaptureVideoPreviewLayer!

func newVideoCaptureSession() throws -> AVCaptureSession? {

// Identify the device that we will be capturing from

let videoCamera = AVCaptureDevice.default(for: AVMediaType.video)

// try to create an input capturer

do {

let videoInput: AVCaptureDeviceInput? = try AVCaptureDeviceInput.init(device: videoCamera!)

let captureSession = AVCaptureSession()

// we are going to ask for it to handle the data that comes in from the videoInput

captureSession.addInput(videoInput!)

return captureSession

} catch let error as NSError { // catch an error if we couldn't capture input

throw error

}

}

func addPreviewLayerForSession(session: AVCaptureSession) {

// all views are backed by a layer

// we only use the layer for special circumstances like using it to create a preview layer

let rootLayer = self.view.layer

// create a preview layer that can display the video stream coming in from captureSession

previewLayer = AVCaptureVideoPreviewLayer(session: session)

// set its frame to be the same as rootLayer's frame to take up the whole screen

previewLayer.frame = rootLayer.frame

// add this previewLayer as a sub layer of rootLayer to display video stream

rootLayer.addSublayer(previewLayer)

}

override func viewDidLoad() {

super.viewDidLoad()

do {

try captureSession = newVideoCaptureSession()

// create a preview layer that will display the data that this captureSession gets

addPreviewLayerForSession(session: captureSession!)

// start camera

captureSession!.startRunning()

} catch let error as NSError {

fatalError("Error capturing video \(error)")

}

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

// after the view appears then start camera

captureSession?.startRunning()

}

override func viewDidDisappear(\_ animated: Bool) {

// stop camera then have the view disappear

captureSession?.stopRunning()

super.viewDidDisappear(animated)

}

func prefersStatusBarHidden() -> Bool {

return true

}

func CGRectMake(\_ x: CGFloat, \_ y: CGFloat, \_ width: CGFloat, \_ height: CGFloat) -> CGRect {

return CGRect(x: x, y: y, width: width, height: height)

}

override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) {

// change the previewLayer's frame to match the new frame size after orientation change

previewLayer.frame = CGRectMake(0, 0, size.width, size.height)

// figure out the current orientation of the device

let currentOrientation = UIDevice.current.orientation

// create an instance of AVCaptureVideoOrientation

let videoOrientation = AVCaptureVideoOrientation(rawValue: currentOrientation.rawValue)!

// then tell the previewLayer how it should change itself

self.previewLayer.connection!.videoOrientation = videoOrientation

}

}