



Praktikum – iOS-Entwicklung

Sommersemester 2018

Prof. Dr. Linnhoff-Popien

Markus Friedrich, Kyrill Schmid

Sensors

Core Motion, Core Location, Multimedia

Core Motion

Framework, das für das Auslesen von...

- ...Beschleunigungssensoren
- ...Gyroskopen
- ...Pedometern
- ...anderen Ereignissen mit Umweltbezug

zuständig ist.

Wichtig: Sensordaten können in „rohem“ Zustand, oder vorverarbeitet abgerufen werden.

Core Motion – Basissensoren

Im `CMMotionManager` stehen die folgenden Sensoren zur Verfügung:

- **Beschleunigungssensor**
- **Gyroskop**
- **Magnetometer**
- **Device Motion** (Bereits vorverarbeitete Werte aus Sensorfusion)

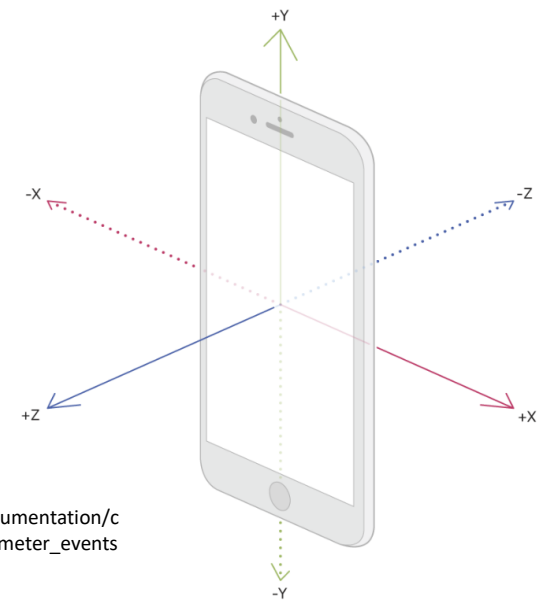
Wichtig: Pro App nur eine Instanz von `CMMotionManager` initialisieren!

Wichtig: Ab iOS 10 muss in der `Info.plist` Datei Beschreibungen für verwendete Daten (Schlüssel) aufgeführt werden.

```
<key>NSMotionUsageDescription</key>  
<string>In order to do cool stuff I  
need access to your motion data.  
</string>
```

Core Motion – Beschleunigungssensor

- Beschleunigung in drei Richtungen
- Skalierung: 1.0 entspricht 9.8 m/s^2



```
let mm = CMMotionManager()

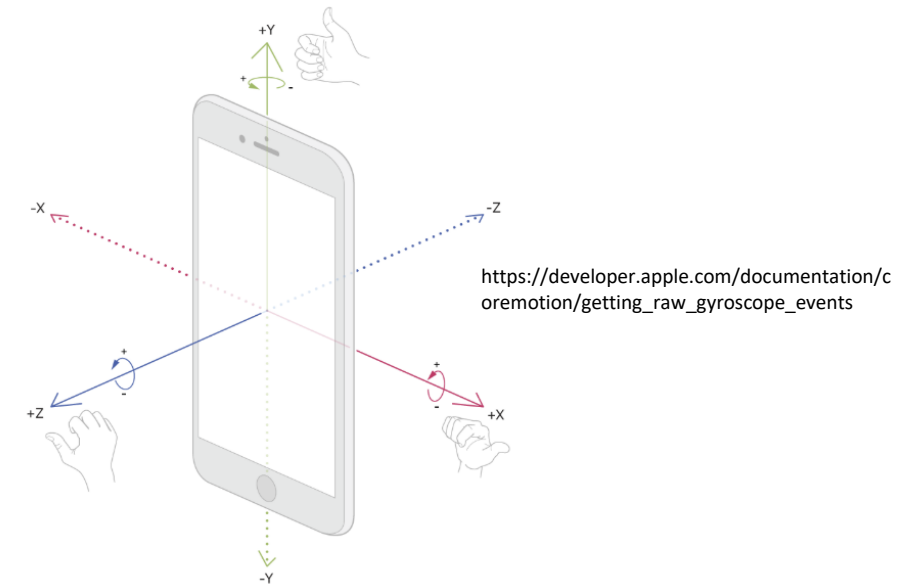
mm.accelerometerUpdateInterval = 1.0 / 60.0//60Hz
mm.startAccelerometerUpdates(to: OperationQueue.current!, withHandler: { data, error in

    guard error == nil else { return }
    guard let accelerometerData = data else { return }

    print(accelerometerData.acceleration.x)
})
```

Core Motion – Gyroskop

- Drehgeschwindigkeit um die drei Achsen
- Einheit: Winkel (in Rad) pro Sekunde
- Richtung: Vorzeichen



```
let mm = CMMotionManager()

mm.gyroUpdateInterval = 1.0 / 60.0//60Hz
mm.startGyroUpdates(to: OperationQueue.current!, withHandler: { data, error in

    guard error == nil else { return }
    guard let gyroData = data else { return }

    print(gyroData.rotationRate.x)
})
```

Core Motion – Magnetometer

- Misst das Magnetfeld der Erde relativ zum Gerät in drei Richtungen
- Einheit: Mikrotelas
- Richtung: Vorzeichen
- Code ist analog zu den anderen Sensoren.
- Datentyp: `CMagnetometerData` enthält Eigenschaft `magneticField` vom Typ `CMMagneticField(x,y,z)`

Core Motion – Device Motion

Liefert **vorverarbeitete Daten** (entfernt z.B. Einflüsse der Gravitation auf Gyroskopdaten).

Folgende Daten stehen zur Verfügung (aus `CMDeviceMotion`):

- **Attitude** (Orientierung relativ zu einer Referenzpose)
- **rotationRate** (Rotationsrate ohne Bias)
- **Gravity** (Gravitationsvektor)
- **userAcceleration** (Beschleunigung ohne Gravitation)
- **magneticField** (Kalibriertes Magnetfeld)
- **heading** (Neigung relativ zu einer Referenzpose)

Core Motion – Device Motion

Beispiel:

```
let mm = CMMotionManager()

mm.deviceMotionInterval = 1.0 / 60.0//60Hz
mm.showsDeviceMovementDisplay = true // Zeigt standard Kalibrierungsinterface
mm.startDeviceMotionUpdates(to: OperationQueue.current!, withHandler: { data, error in

    guard error == nil else { return }
    guard let motionData = data else { return }

    print(motionData.userAcceleration.x)
})
```

Core Motion – Verfügbarkeit abfragen

`CMMotionManager` besitzt Eigenschaften, mit denen man die Verfügbarkeit bestimmter Sensoren abfragen kann:

```
let mm = CMMotionManager()

print(„Gyroscope: \ (mm.isGyroAvailable)“);
print(„Accelerometer: \ (mm.isAccelerometerAvailable)“);
print(„Magnetometer: \ (mm.isMagnetometerAvailable)“);
print(„Device Motion: \ (mm.isDeviceMotionAvailable)“);
```

Core Motion – Motion Activity Manager

- Abfrage von **high-level Bewegungsdaten** (stationary | walking | running | automotive | cycling + Konfidenz (low | medium | high) + Startdatum)

- Beispiel:

```
let mam = CMMotionActivityManager()

mam.startActivityUpdates(to: OperationQueue.current!, withHandler: { data, error
in
    guard error == nil else { return }
    guard let activityData = data else { return }

    print(activityData.cycling)
})
mam.stopActivityUpdates()
```

- Historische Daten: `queryActivityStarting(from, to, handler)`

Core Motion – Weitere Sensoren

Neben den im `CMMotionManager` im angebotenen Sensoren existieren noch weitere:

- **Pedometer** (Schrittzähler)
- **Altimeter** (Höhenlage)

Authorisierung:

Pedometer und Altimeter (wie `CMMotionActivityManager`) verfügen über die Methode `authorizationStatus()`, mit der man überprüfen kann, ob für die App der Zugriff auf den jeweiligen Sensor erlaubt ist.

Core Motion – Pedometer

- Die Klasse `CMPedometer` zählt Schritte.
- **Unterscheidung:** `startEventUpdates()`, `startUpdates()`
 - Datentyp: `CMPedometerData` enthält:
 - `startDate`, `endDate`
 - `numberOfSteps`
 - `Distance`
 - `averageActivePace`
 - `currentPace`
 - `currentCadence`
 - `floorsAscended`
 - `floorsDescended`
 - Datentyp: `CMPedometerEvent` enthält Datum und Eventtyp (pause | resume)
- Historische Daten: `queryPedometerData(from, to, handler)`

Wichtig zu prüfen:

`Is{StepCounting | Distance | FloorCounting | Pace | Cadence | PedometerEventTracking}Available()`

Core Motion – Pedometer

Beispiel:

```
let pedometer = CMPedometer()

pedometer.startUpdates(from: Date()) { pedometerData, error in
    guard let pedometerData = pedometerData, error == nil else { return }
    if pedometer.isDistanceAvailable() {
        print(pedometerData.distance)
    }
}

pedometer.startEventUpdates() { pedometerEventData, error in
    guard let pedometerEventData = pedometerEventData, error == nil else { return }
    print(pedometerEventData.type)
}

pedometer.stopUpdates()
pedometer.stopEventUpdates()
```

Core Motion – Altimeter

Zeigt Änderungen in der Höhenlage an.

Beispiel:

```
let altimeter = CMAltimeter()
if altimeter.isRelativeAltitudeAvailable() == false {return}

altimeter.startRelativeAltitudeUpdates(to: OperationQueue.current!, withHandler:
{ data, error in

    guard error == nil else { return }
    guard let altitudeData = data else { return }

    print(altitudeData.relativeAltitude)
})

altimeter.stopRelativeAltitudeUpdates()
```

Core Location

- Das Core Location Framework beinhaltet die folgenden **Location Services**:

Service	Beschreibung
Standard Location Service	Präzision ++, Frequenz ++, Batterie --
Significant-change Location Service	Präzision o, Frequenz -, Batterie +
Visits Service	Präzision -, Frequenz --, Batterie ++ => Nicht für Echtzeit Anwendungen, sondern für das Finden von Benutzeraktivitätsmuster.
Region Monitoring	Nachricht, wenn Benutzer Geo-Region betritt
iBeacon Ranging	Nachricht, wenn Benutzer i. d. Nähe v. BT Beacon
Heading Service	Kompass (basierend auf Magnetometerdaten)
Geocoding Services	Position => „Placemark“ (Adresse), auch „reverse“

- Die Dienste nutzen die **komplette verfügbare Sensorik** (GPS, Wi-Fi, Bluetooth, Magnetometer,...)

Core Location - Authorisierungsmodi

Festlegung, **wann** eine App **welchen** Location Service benutzen darf.

Service	When-In-Use	Always
Standard Location Service	Ja	Ja
Significant-change Location Service	Nein	Ja
Visits Service	Nein	Ja
Region Monitoring	Nein	Ja
iBeacon Ranging	Ja	Ja
Heading Service	Ja	Ja
Geocoding Services	Ja	Ja

When-in-use:

- Kann LSs nur aus Vordergrund heraus starten.
- Location Event => App wird **nicht** gestartet (aber aus suspended)

Always:

- Kann LSs aus Vorder- und Hintergrund heraus starten.
- Location Event => App wird gestartet.

Core Location – When-In-Use Authorisierung

Beispiel:

Wichtig: Info.plist => NSLocationWhenInUseUsageDescription

```
let locationManager = CLLocationManager()
func enableBasicLocationServices() { // Methode einer Klasse, die CLLocationManagerDelegate
                                    // implementiert
    locationManager.delegate = self
    switch CLLocationManager.authorizationStatus() {
    case .notDetermined:
        locationManager.requestWhenInUseAuthorization()
        break
    case .restricted, .denied:
        disableMyLocationBasedFeatures()
        break
    case .authorizedWhenInUse, .authorizedAlways:
        enableMyWhenInUseFeatures()
        break
    }
}
```

Wichtig: Da sich die Verfügbarkeit der Location Services laufend ändern kann, ist es wichtig, **func locationManager(_ manager: CLLocationManager, didChangeAuthorization status: CLErrorAuthorizationStatus)** zu implementieren

https://developer.apple.com/documentation/corelocation/choosing_the_authorization_level_for_location_services/requesting_when_in_use_authorization

Core Location – Benutzung des Standard LS

Beispiel (1): Starten eines LS:

https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_standard_location_service

```
let lm = CLLocationManager()
func startReceivingLocationChanges() { // Methode einer Klasse, die CLLocationManagerDelegate
                                     // implementiert
    let authorizationStatus = CLLocationManager.authorizationStatus()
    if authorizationStatus != .authorizedWhenInUse && authorizationStatus != .authorizedAlways {
        return
    }
    if !CLLocationManager.locationServicesEnabled() {
        return
    }

    lm.desiredAccuracy = kCLLocationAccuracyHundredMeters // Energieverbrauch!
    lm.distanceFilter = 100.0 // In meters.
    lm.delegate = self
    lm.startUpdatingLocation()
}
```

Core Location – Benutzung des Standard LS

Beispiel (2): Verarbeitung neuer Werte

https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_standard_location_service

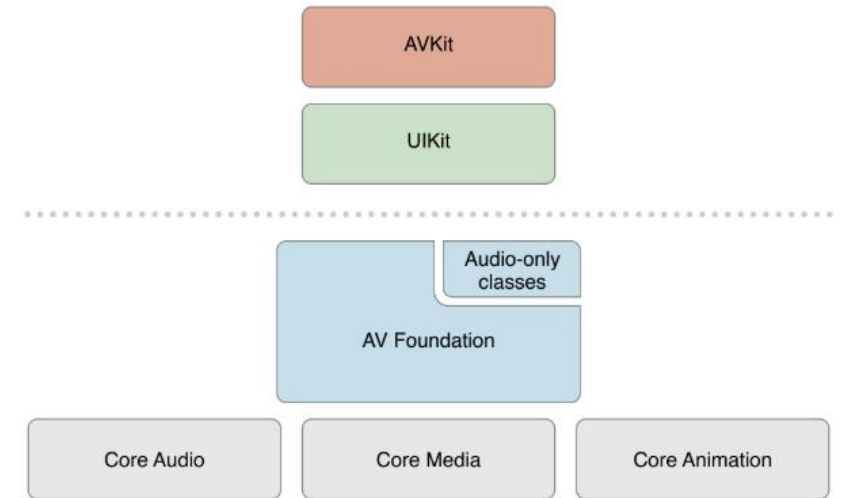
```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations:
[CLLocation]) { // Methode einer Klasse, die CLLocationManagerDelegate
                // implementiert.

    let lastLocation = locations.last!

    print("Most recent location: \(lastLocation).")
}
```

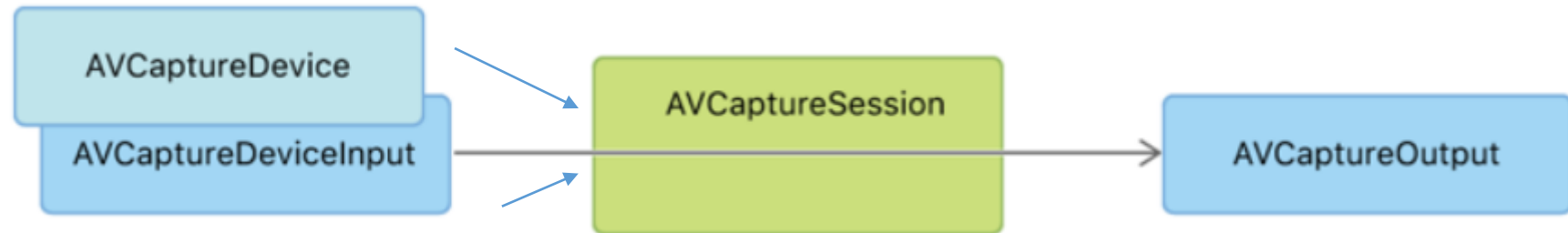
Multimedia

- Natürlich lassen sich auch Videos, Fotos und Audio aufnehmen
- Zentral ist hierbei das Framework **AVFoundation**.
- Wir konzentrieren uns auf den Low-Level Zugriff auf Rohdaten.



https://developer.apple.com/library/content/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html

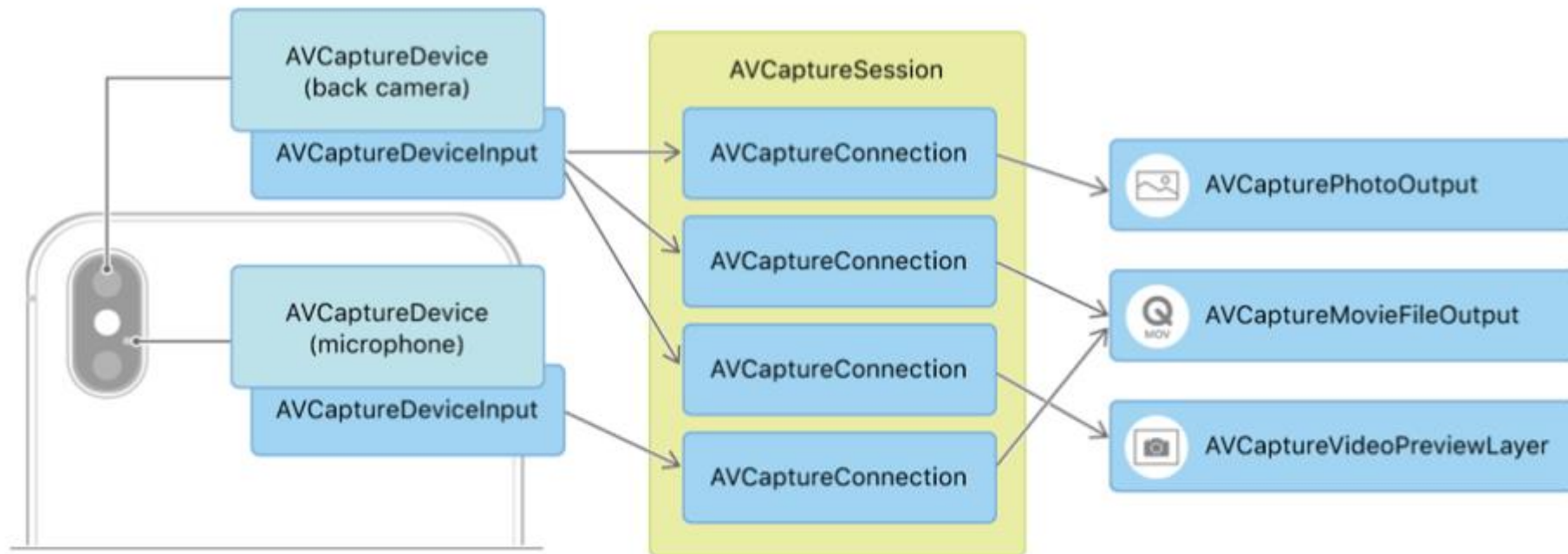
- Elemente:



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

Multimedia – Aufsetzen einer Capture Session

Eine Session verbindet Input Geräte (AVCaptureInput) mit Output-Objekten (AVCaptureOutput):



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

Multimedia – Aufsetzen einer Capture Session

Beispiel (1): Aufsetzen einer Video Capture Session:

```
var captureSession = AVCaptureSession()

captureSession.beginConfiguration()

let videoDevice = AVCaptureDevice.default(.builtInWideAngleCamera,
                                          for: .video, position: .unspecified)

guard
    let videoDeviceInput = try? AVCaptureDeviceInput(device: videoDevice!),
    captureSession.canAddInput(videoDeviceInput)
else { return }

captureSession.addInput(videoDeviceInput)
```

Multimedia – Aufsetzen einer Capture Session

Beispiel (2): Weiterführend

```
let cameraQueue = dispatch_queue_create("cameraQueue", DISPATCH_QUEUE_SERIAL)

videoCaptureOutput.setSampleBufferDelegate(myDelegate, queue: cameraQueue)

captureSession.addOutput(videoCaptureOutput)

captureSession.commitConfiguration()

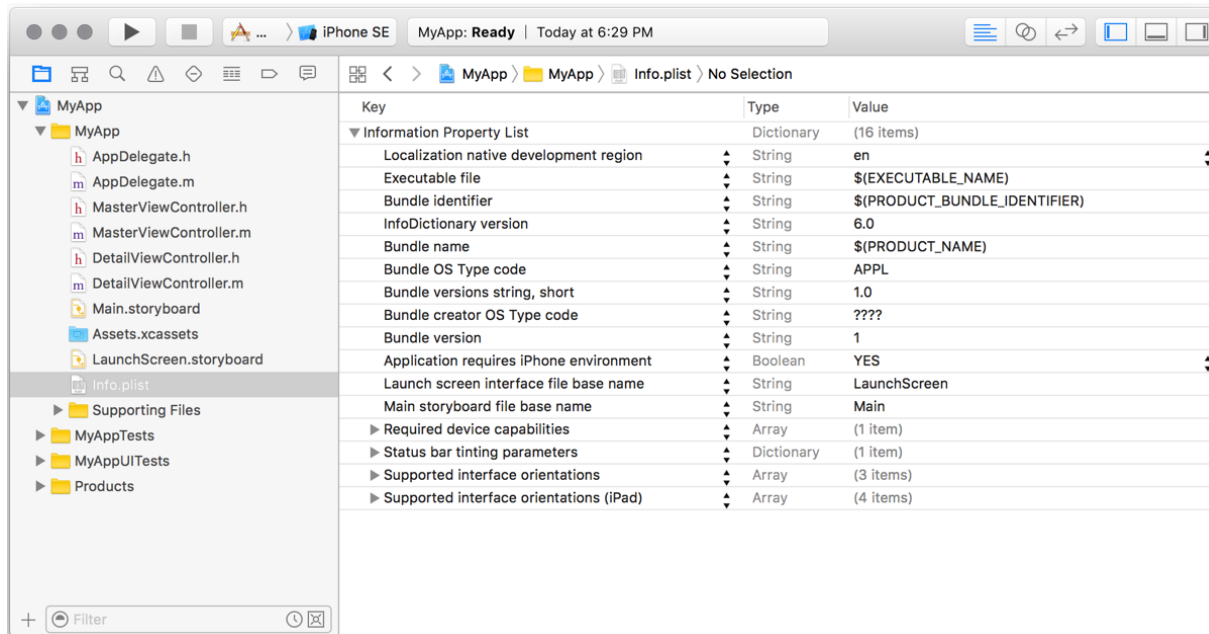
captureSession.startRunning()
```

myDelegate ist vom Typ AVCaptureVideoDataOutputSampleBufferDelegate.

Wichtig: func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from connection: AVCaptureConnection)

Multimedia - Berechtigungen einholen

Für den Zugriff auf Kamera und Mikrofon, muss die Info.plist Datei editiert werden



Schlüssel:

NSCameraUsageDescription,
NSMicrophoneUsageDescription

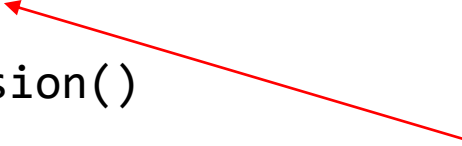
Schlüssel enthalten Erklärungen, warum Gerät genutzt werden soll.

```
<key>NSCameraUsageDescription</key>  
<string>In order to do cool stuff I need  
access to your camera.  
</string>
```

<https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/AboutInformationPropertyListFiles.html>

Multimedia – Berechtigungen prüfen

```
switch AVCaptureDevice.authorizationStatus(for: .video) {  
    case .authorized: // The user has previously granted access to the camera.  
        self.setupCaptureSession()  
  
    case .notDetermined: // The user has not yet been asked for camera access.  
        AVCaptureDevice.requestAccess(for: .video) { granted in  
            if granted {  
                self.setupCaptureSession()  
            }  
        }  
  
    case .denied: // The user has previously denied access.  
        return  
    case .restricted: // The user can't grant access due to restrictions.  
        return  
}
```



asynchron!

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