

# Sensors

## Lecture 7

CIS 1951

# Last time, in CIS 1951...

## Custom Views & Event Handling

- GeometryReader, safe area
- SwiftUI shapes, .fill/.stroke, .clipShape
- Understanding event propagation and handling
- Keyboard handling and text input events
- Custom gesture recognition in SwiftUI
- **Questions? Comments? Feedback?**

# CIS 1951 as a whole

Lectures 1-6: The Basics

**Lectures 7-10: Technologies**

Lectures 11-13: Beyond Development

# The iPhone



## Sensors

Face ID  
LiDAR Scanner  
Barometer  
High dynamic range gyro  
High-g accelerometer  
Proximity sensor  
Dual ambient light sensors

## Location

Precision dual-frequency GPS (GPS, GLONASS, Galileo, QZSS, BeiDou, and NavIC)  
Digital compass  
Wi-Fi  
Cellular  
iBeacon microlocation

Camera	iPhone 15 Pro	iPhone 15 Pro Max
	<p><b>Pro camera system</b> 48MP Main: 24 mm, <math>f/1.78</math> aperture, second-generation sensor-shift optical image stabilization, 100% Focus Pixels, support for super-high-resolution photos (24MP and 48MP) 12MP Ultra Wide: 13 mm, <math>f/2.2</math> aperture and <math>120^\circ</math> field of view, 100% Focus Pixels 12MP 2x Telephoto (enabled by quad-pixel sensor): 48 mm, <math>f/1.78</math> aperture, second-generation sensor-shift optical image stabilization, 100% Focus Pixels 12MP 3x Telephoto: 77 mm, <math>f/2.8</math> aperture, optical image stabilization 3x optical zoom in, 2x optical zoom out, 6x optical zoom range Digital zoom up to 15x</p>	<p><b>Pro camera system</b> 48MP Main: 24 mm, <math>f/1.78</math> aperture, second-generation sensor-shift optical image stabilization, 100% Focus Pixels, support for super-high-resolution photos (24MP and 48MP) 12MP Ultra Wide: 13 mm, <math>f/2.2</math> aperture and <math>120^\circ</math> field of view, 100% Focus Pixels 12MP 2x Telephoto (enabled by quad-pixel sensor): 48 mm, <math>f/1.78</math> aperture, second-generation sensor-shift optical image stabilization, 100% Focus Pixels 12MP 5x Telephoto: 120 mm, <math>f/2.8</math> aperture, 3D sensor-shift optical image stabilization and autofocus, tetraprism design 5x optical zoom in, 2x optical zoom out, 10x optical zoom range Digital zoom up to 25x</p>
	<p><b>Both models</b> Customizable default lens (Main) Sapphire crystal lens cover Adaptive True Tone flash Photonic Engine Deep Fusion Smart HDR 5 Next-generation portraits with Focus and Depth Control Portrait Lighting with six effects Night mode Night mode portraits enabled by LiDAR Scanner Panorama (up to 63MP) Photographic Styles Macro photography Apple ProRAW Wide color capture for photos and Live Photos Lens correction (Ultra Wide) Advanced red-eye correction Auto image stabilization Burst mode Photo gislagging Image formats captured: HEIF, JPEG, and DNG</p>	

## Video Recording

4K video recording at 24 fps, 25 fps, 30 fps, or 60 fps  
1080p HD video recording at 25 fps, 30 fps, or 60 fps  
720p HD video recording at 30 fps  
Cinematic mode up to 4K HDR at 30 fps  
Action mode up to 2.8K at 60 fps  
HDR video recording with Dolby Vision up to 4K at 60 fps  
ProRes video recording up to 4K at 60 fps with external recording  
Log video recording  
Academy Color Encoding System  
Macro video recording, including slo-mo and time-lapse  
Slo-mo video support for 1080p at 120 fps or 240 fps  
Time-lapse video with stabilization  
Night mode Time-lapse  
QuickTake video  
Second-generation sensor-shift optical image stabilization for video (Main)  
Optical image stabilization for video (3x Telephoto)  
3D sensor-shift optical image stabilization and autofocus for video (5x Telephoto)  
Digital zoom up to 9x (iPhone 15 Pro) and 15x (iPhone 15 Pro Max)  
Audio zoom  
True Tone flash  
Cinematic video stabilization (4K, 1080p, and 720p)  
Continuous autofocus video  
Take 8MP still photos while recording 4K video  
Playback zoom  
Video formats recorded: HEVC, H.264, and ProRes  
Stereo recording

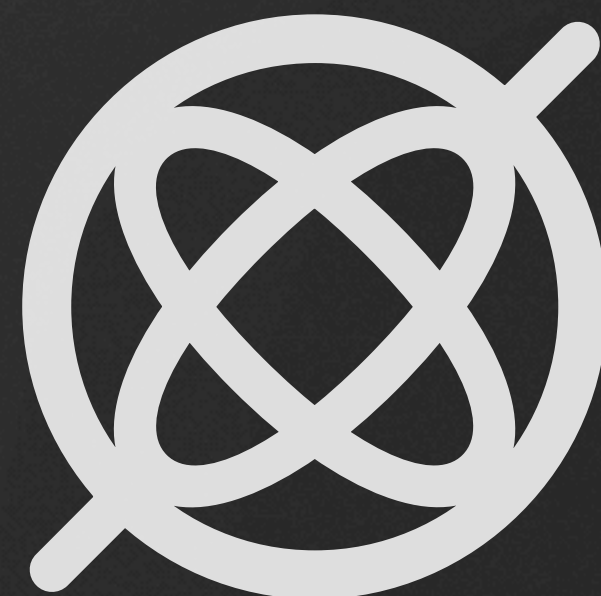
## TrueDepth Camera

12MP camera  
 $f/1.9$  aperture  
Autofocus with Focus Pixels  
Retina Flash  
Photonic Engine  
Deep Fusion  
Smart HDR 5  
Next-generation portraits with Focus and Depth Control  
Portrait Lighting with six effects  
Animoji and Memoji  
Night mode  
Photographic Styles  
Apple ProRAW  
Wide color capture for photos and Live Photos  
Lens correction  
Auto image stabilization  
Burst mode  
4K video recording at 24 fps, 25 fps, 30 fps, or 60 fps  
1080p HD video recording at 25 fps, 30 fps, or 60 fps  
Cinematic mode up to 4K HDR at 30 fps  
HDR video recording with Dolby Vision up to 4K at 60 fps  
ProRes video recording up to 4K at 60 fps with external recording  
Log video recording  
Academy Color Encoding System  
Slo-mo video support for 1080p at 120 fps  
Time-lapse video with stabilization  
Night mode Time-lapse  
QuickTake video  
Cinematic video stabilization (4K, 1080p, and 720p)

# This week



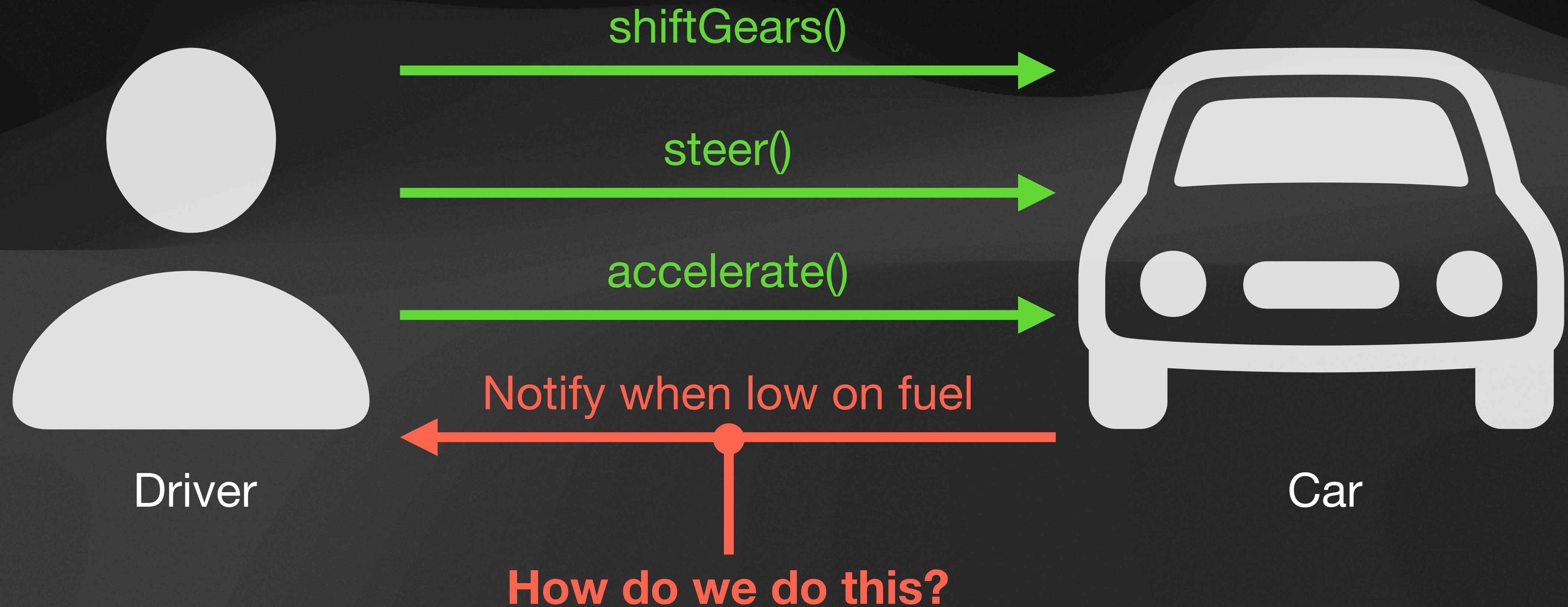
**Location**



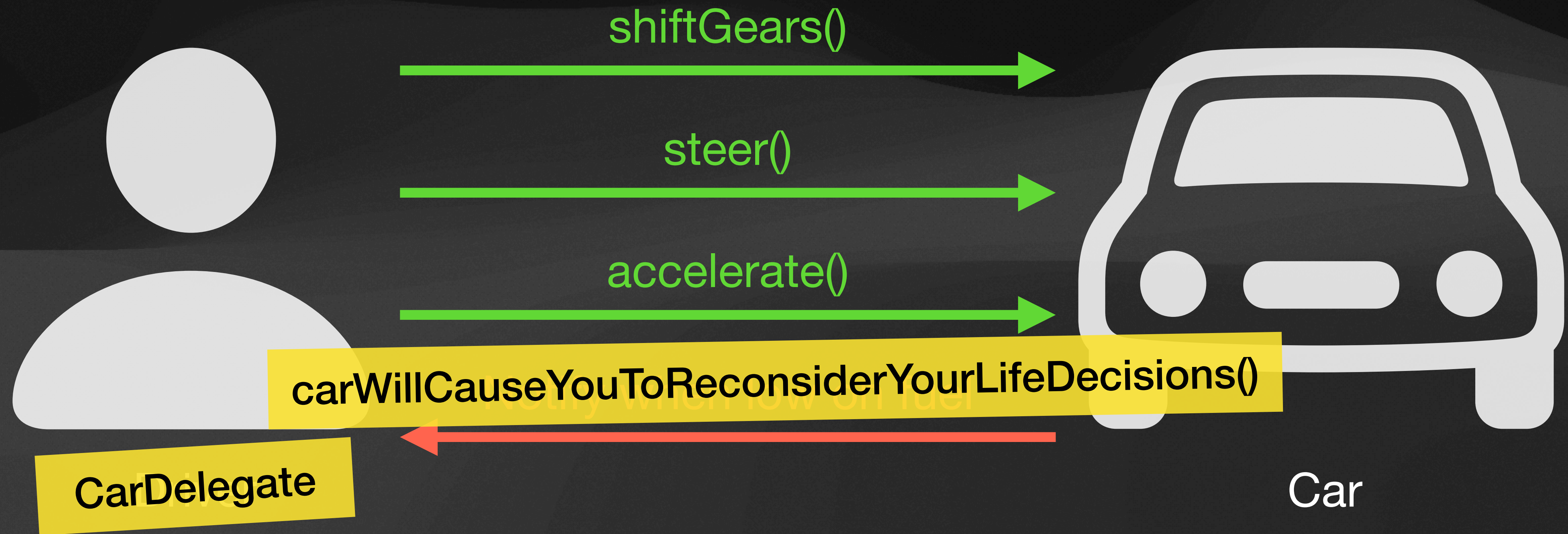
**Motion**

**But first, some design patterns**

Suppose we're designing a **Car** interface...



# The Delegate Pattern





# The Delegate Pattern

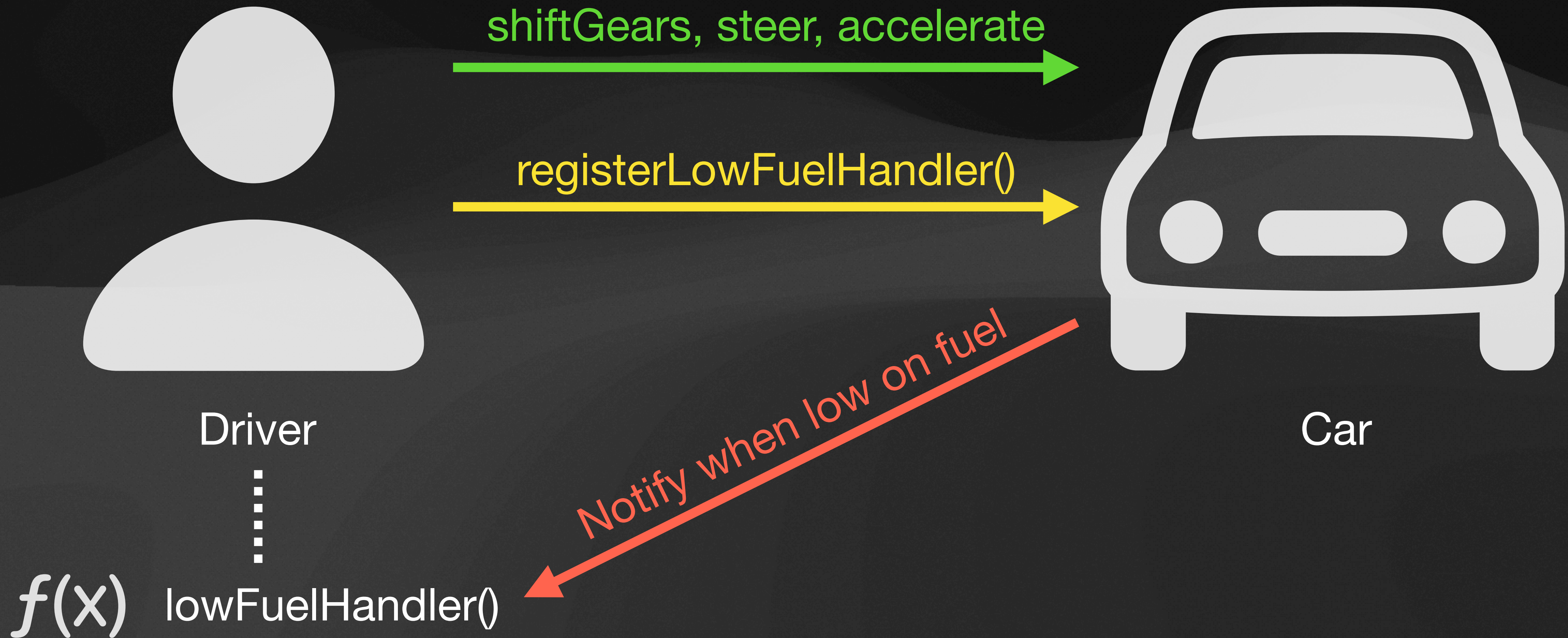
```
protocol CarDelegate: AnyObject {
    func carWillCauseYouToReconsiderYourLifeDecisions()
}

class Car {
    weak var delegate: CarDelegate?

    func shiftGears() {}
    func steer() {}
    func accelerate() {}
}

class Driver: CarDelegate {
    func carWillCauseYouToReconsiderYourLifeDecisions() {
        // Panic...
    }
}
```

# The Observer Pattern



# The Observer Pattern

```
class Car {  
    func shiftGears() {}  
    func steer() {}  
    func accelerate() {}  
  
    func registerLowFuelHandler(_ handler: @escaping () -> Void) {}  
}  
  
let car = Car()  
car.registerLowFuelHandler { ← Similar to .onChange/.onAppear  
    // Panic...  
}
```

# The Observer Pattern

## Another Example

```
class Driver {
  init() {
    let car = Car()
    car.registerLowFuelHandler { [weak self] in
      if let self = self {
        panic()
      }
    }
  }
}

func panic() {}
}
```

[weak self]

if let self = self

???



# CIS 1905: Rust

2/27/2024

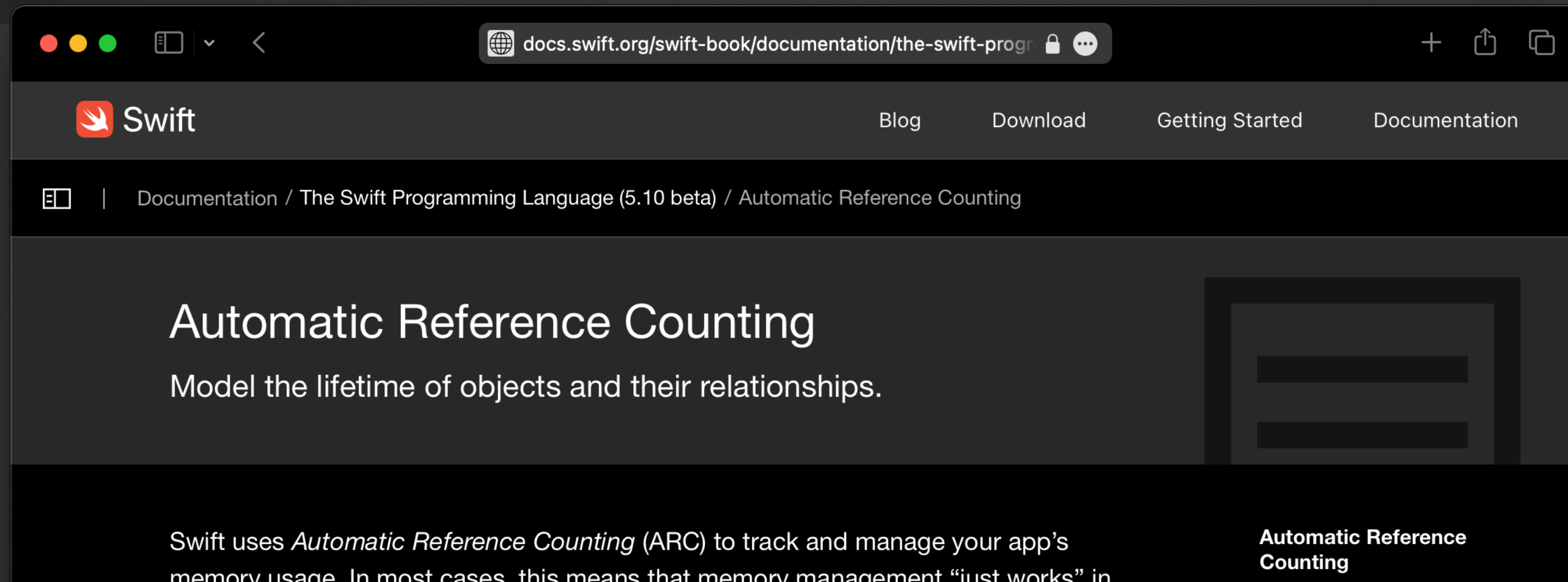
Smart Pointers

15

[weak self]

if let self = self

???



The background features a series of overlapping, wavy shapes in various shades of blue, ranging from a very dark, almost black blue at the top to a vibrant, bright blue at the bottom. The word "Location" is written in a clean, white, sans-serif font, positioned on the left side of the image, overlapping the darker blue waves.

**Location**

developer.apple.com/documentation/corelocation

Developer News Discover Design Develop Distribute Support Account

Core Location Documentation / Core Location Language: Swift API Changes: Show

### Essentials

- Configuring your app to use location services
- Supporting live updates in SwiftUI and Mac Cata...
- > CLLocationManager
- > CLBackgroundActivitySession
- > CLLocationUpdate
- > CLLocationManagerDelegate
- { Adopting live updates in Core Location
- { Monitoring location changes with Core Location

### Authorization

- Requesting authorization to use location services
- > CLAuthorizationStatus
- > CLAccuracyAuthorization
- T CLLocationAlwaysAndWhenInUseUsageDescrip...
- T CLLocationWhenInUseUsageDescription

Filter /

## Framework

# Core Location

Obtain the geographic location and orientation of a device.

iOS 2.0+ iPadOS 2.0+ macOS 10.6+ Mac Catalyst 13.0+ tvOS 9.0+  
watchOS 2.0+ visionOS 1.0+

## Overview

Core Location provides services that determine a device's geographic location, altitude, and orientation, or its position relative to a nearby iBeacon device. The framework gathers data using all available components on the device, including the Wi-Fi, GPS, Bluetooth, magnetometer, barometer, and cellular hardware.

You use instances of the [CLLocationManager](#) class to



# 1 Set up a CLLocationManager

```
import CoreLocation
import SwiftUI

class GenericViewModel {
    let locationManager = CLLocationManager()
}
```

## 2 Set up a delegate

From the Objective-C days

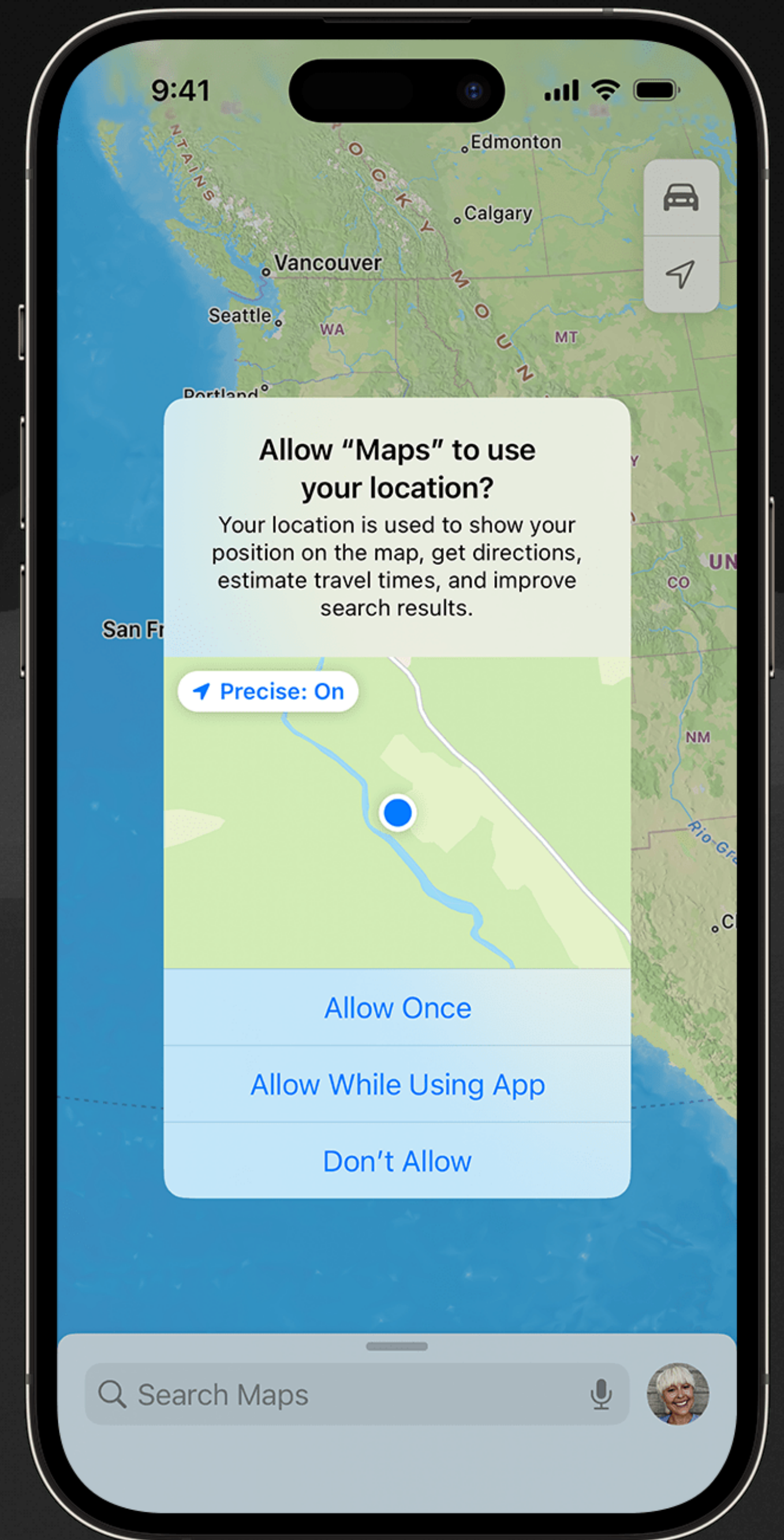


```
class GenericViewModel: NSObject, CLLocationManagerDelegate {  
    let locationManager = CLLocationManager()  
  
    override init() {  
        super.init()  
        locationManager.delegate = self  
    }  
}
```

# 3 Request access

```
locationManager.requestWhenInUseAuthorization()
```

Requires a “purpose string” (*more on that later*)



# 4 Respond to the user's choice

```
class GenericViewModel: NSObject, CLLocationManagerDelegate {
    // ...

    func locationManagerDidChangeAuthorization(_ manager: CLLocationManager) {
        switch manager.authorizationStatus {
        case .authorizedWhenInUse, .authorizedAlways:
            doSomethingWithLocationAccess()
        case .denied, .restricted:
            showAlert("Enyabwe wocation access in settings pwease 🙄")
        default:
            break
        }
    }
}
```

## 5 Request the user's location

```
func doSomethingWithLocationAccess() {  
    locationManager.requestLocation()  
}
```

## 6 Receive the user's location

```
func locationManager(_ manager: CLLocationManager,  
                    didUpdateLocations locations: [CLLocation]) {  
    if let location = locations.last {  
        send(location, to: sketchyServer)  
    }  
}
```

# 7 Handle errors

```
func locationManager(_ manager: CLLocationManager,
                    didFinishWithError error: Error) {
    showAlert("Owoh noes, can't find yow wocation!")
}
```

App will crash if this is not present!

# Other things you can do with CoreLocation

...that we won't have time to cover

- Continuous location updates
- Significant location change notifications
- Geofencing
- Background access
- And much more — **check the docs!**



# Motion

developer.apple.com/documentation/coremotion

Developer News Discover Design Develop Distribute Support Account

Core Motion Documentation / Core Motion Language: Swift API Changes: Show

### Essentials

- Core Motion updates
- > CMMotionManager

### Device motion

- Getting processed device-motion data
- > CMDeviceMotion
- > CMAttitude
- > CMAttitudeReferenceFrame
- > CMHeadphoneMotionManager

### Accelerometers

- Getting raw accelerometer events
- > CMAccelerometerData
- > CMRecordedAccelerometerData
- > CMSensorRecorder
- CMSensorDataList

Filter /

## Framework

# Core Motion

Process accelerometer, gyroscope, pedometer, and environment-related events.

iOS 4.0+ iPadOS 4.0+ macOS 10.15+ Mac Catalyst 13.0+  
watchOS 2.0+ visionOS 1.0+

## Overview

Core Motion reports motion- and environment-related data from the available onboard hardware of iOS, iPadOS, watchOS, and visionOS devices. This hardware includes the device's accelerometers and gyroscopes, and, when available, the pedometer, magnetometer, and barometer. Use this data in your app as input for user interactions, fitness tracking, health-related matters, and more. For example, a game might use accelerometer

# 1 Set up a CMMotionManager

```
import CoreMotion  
let motionManager = CMMotionManager()
```

## 2 Check if motion data is available

```
if locationManager.isDeviceMotionAvailable {  
    // TODO: Start requesting the device's motion  
} else {  
    showAlert("Looks like device motion is not available >w<")  
}
```

## 3 Subscribing to device motion data

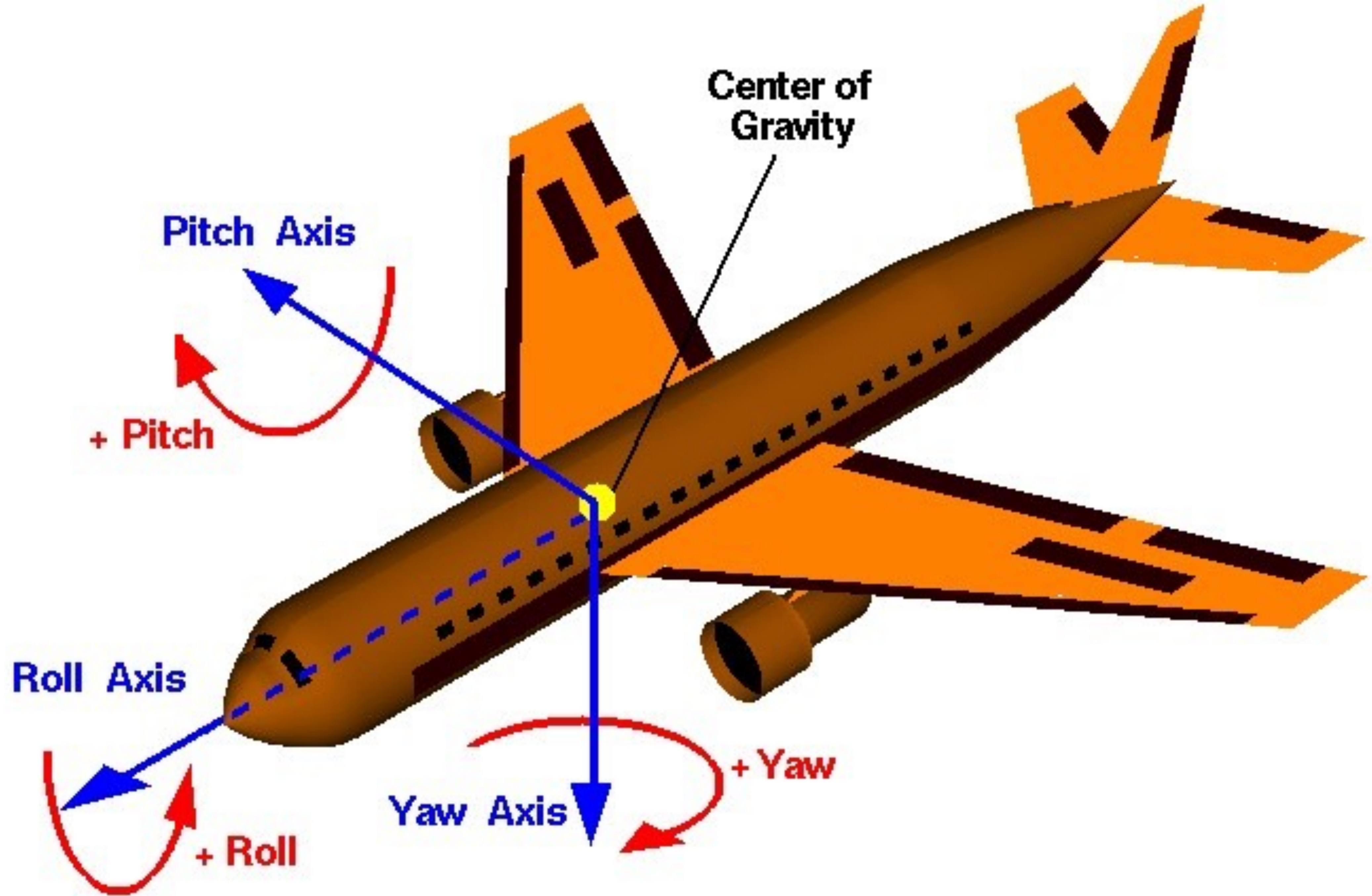
```
motionManager.deviceMotionUpdateInterval = 1 / 60
motionManager.startDeviceMotionUpdates(to: .main) { motion, error in
    if let motion {
        send(motion, to: sketchyServer)
    } else if let error {
        showAlert("oopsie whoopsie, something's wrong nya~")
    }
}
```

You can also do this without a handler function — check the docs

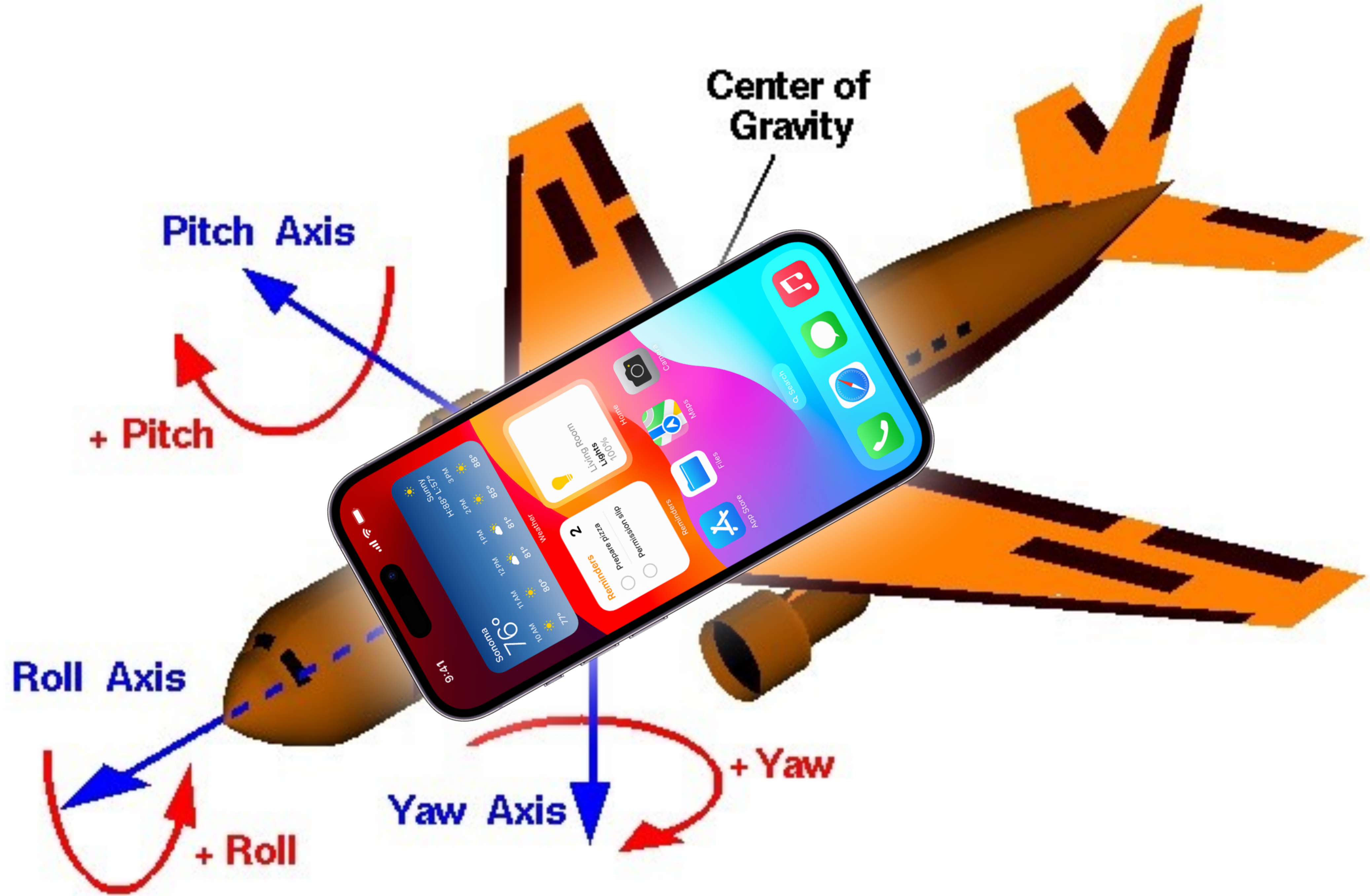
## 4 Clean up

```
class GenericViewModel {  
    // ...  
  
    deinit {  
        motionManager.stopDeviceMotionUpdates()  
    }  
}
```

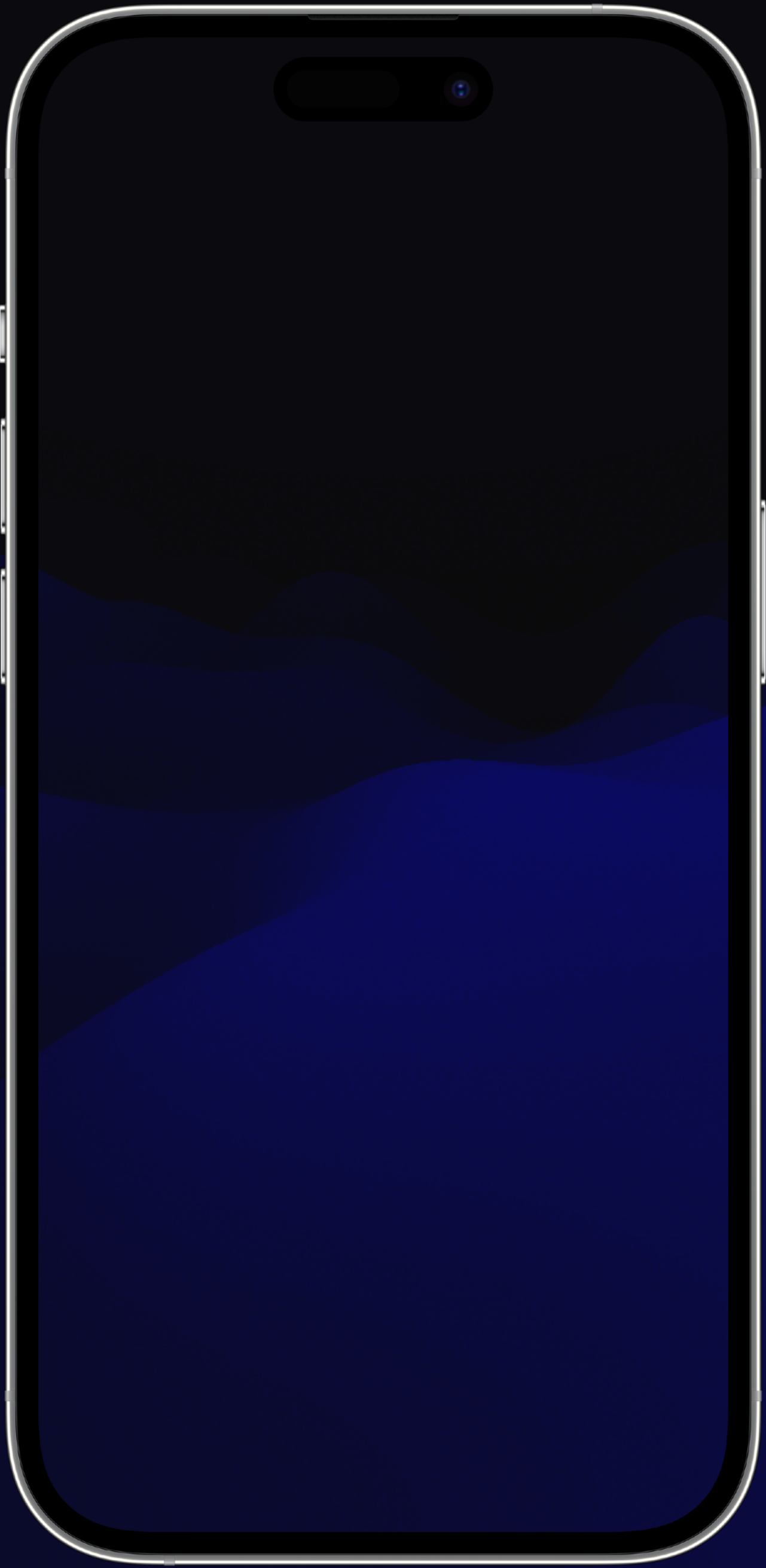
# Attitude



# Attitude







# What data can you get?

- Device rotation (“attitude”)
- Rotation rate
- Gravity
- Device acceleration
- Magnetic fields
- Compass heading
- Headphone motion
- Raw gyroscope and accelerometer data
- Altitude
- Step count
- Pressure and temperature
- Movement disorder symptom data (Apple Watch only)
- Water temperature and depth (Apple Watch Ultra only)
- Fall detection events (Apple Watch only)

Some of these require **other methods** in CoreMotion

# Purpose Strings

# Purpose Strings

*Required* for privacy-sensitive features to work

The screenshot shows the Xcode interface for a project named 'Guesstaurant'. The 'Info' tab is selected and highlighted with a blue circle. The 'Custom iOS Target Properties' table is visible, with a red arrow pointing to the 'Privacy - Motion Usage Description' property.

Key	Type	Value
Bundle name	String	\$(PRODUCT_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Application supports indirect in...	Boolean	YES
Requires Full Screen	Boolean	YES
Bundle version	String	\$(CURRENT_PROJECT_VERSION)
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone en...	Boolean	YES
> Application Scene Manifest	Dictionary	(2 items)
> Supported interface orientations	Array	(2 items)
Privacy - Motion Usage Descri...	String	We use your motion so you can indicate whether you got an answer correct or not.
Privacy - Location When In Us...	String	We use your location to select restaurants close to you.
Bundle OS Type code	String	\$(PRODUCT_BUNDLE_PACKAGE_TYPE)
> Maps routing app supported m...	Array	(0 items)
> Launch Screen	Dictionary	(1 item)
Default localization	String	\$(DEVELOPMENT_LANGUAGE)
Bundle version string (short)	String	\$(MARKETING_VERSION)

# Purpose Strings

***Required*** for privacy-sensitive features to work

Specify `NSMotionUsageDescription` and `NSLocationWhenInUseUsageDescription` in `Info.plist`

Privacy - Motion Usage Descri...	⇅	String	We use your motion so you can indicate whether you got an answer correct or not.
Privacy - Location When In Us...	⇅	String	We use your location to select restaurants close to you.



**Allow "Guesstaurant" to use  
your location?**

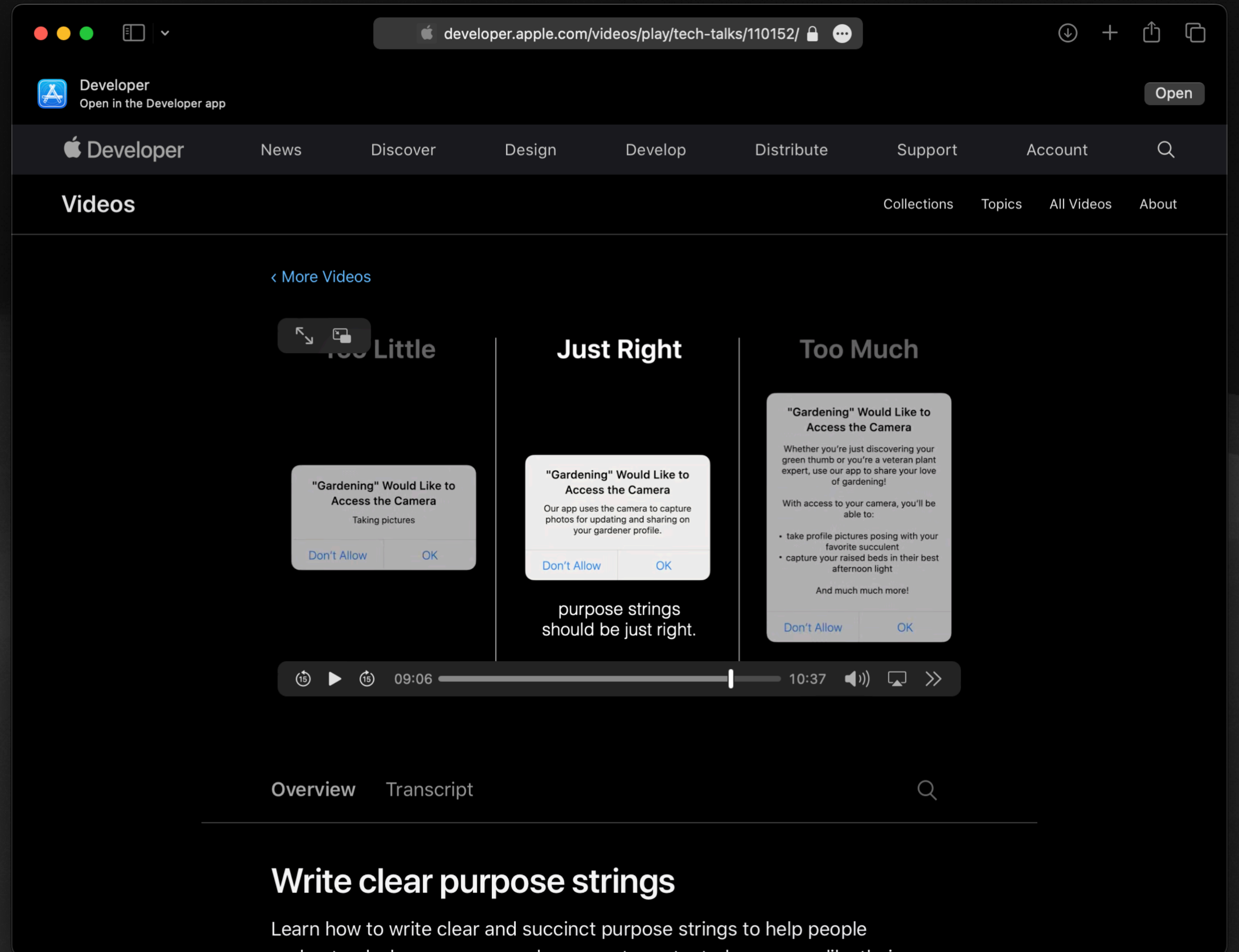
We use your location to select  
restaurants close to you.

# Purpose Strings Must be clear

## Adhere to the requirements for purpose strings

To give people useful, concise information about why you're requesting access to protected resources, make sure each purpose string you provide is valid by checking the following:

- The purpose string isn't blank and doesn't consist solely of whitespace characters.
- The purpose string is shorter than 4,000 bytes. Typical purpose strings are one complete sentence, but you can provide additional information to help a person make the right decision about sharing personal information.
- The purpose string has the proper type that the corresponding key requires, typically a string.
- The purpose string provides a description that's accurate, meaningful, and specific about why the app needs to access the protected resource.



The screenshot shows a video player interface from developer.apple.com. The video content is divided into three vertical panels illustrating different levels of clarity for a camera access purpose string:

- Too Little:** Shows a dialog box with the title "Gardening" Would Like to Access the Camera" and a single line of text: "Taking pictures".
- Just Right:** Shows a dialog box with the same title and a more descriptive sentence: "Our app uses the camera to capture photos for updating and sharing on your gardener profile."
- Too Much:** Shows a dialog box with the same title, followed by a paragraph of text: "Whether you're just discovering your green thumb or you're a veteran plant expert, use our app to share your love of gardening!" and a bulleted list of features: "With access to your camera, you'll be able to: • take profile pictures posing with your favorite succulent • capture your raised beds in their best afternoon light".

Below the panels, a video player control bar shows a progress bar at 09:06. At the bottom of the page, a section titled "Write clear purpose strings" is visible, with the text "Learn how to write clear and succinct purpose strings to help people understand why your app needs access to protected resources like their..."

Unclear purpose strings are a **common App Store rejection!**

**Running your app on-device**

# 1 Plug your device into your computer

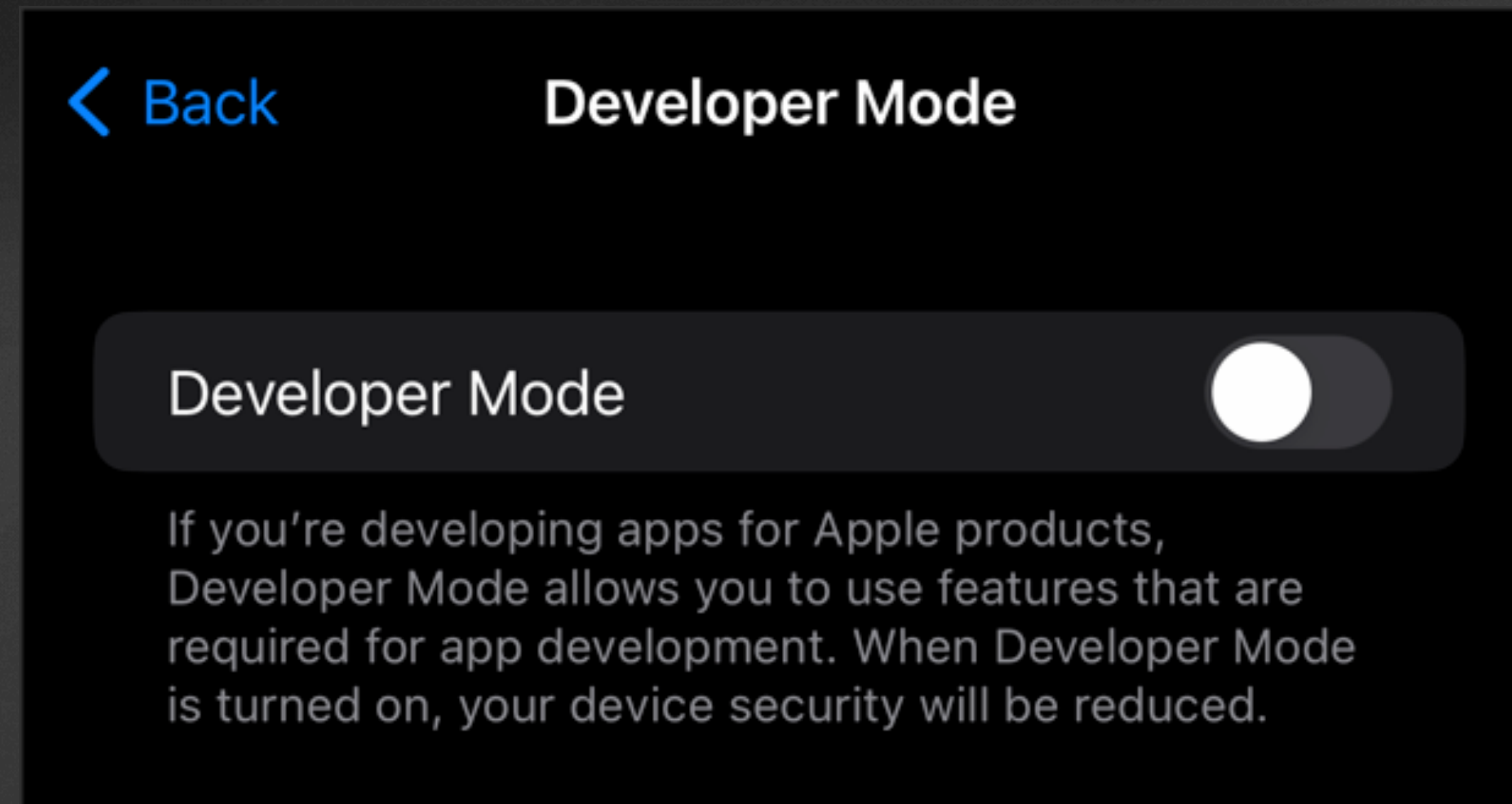


(DALL-E isn't great at this lol)

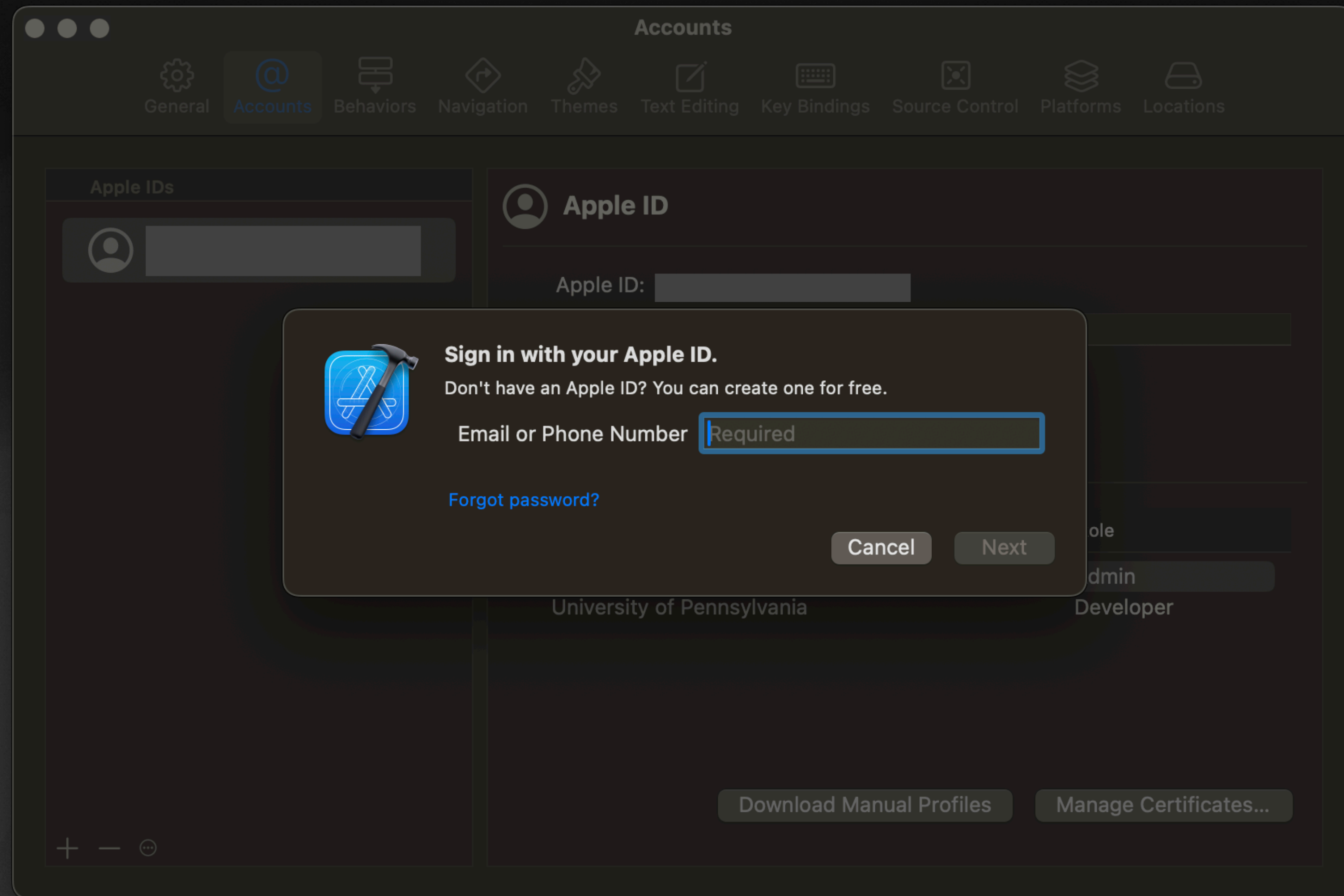


# 2 Enable Developer Mode on your device

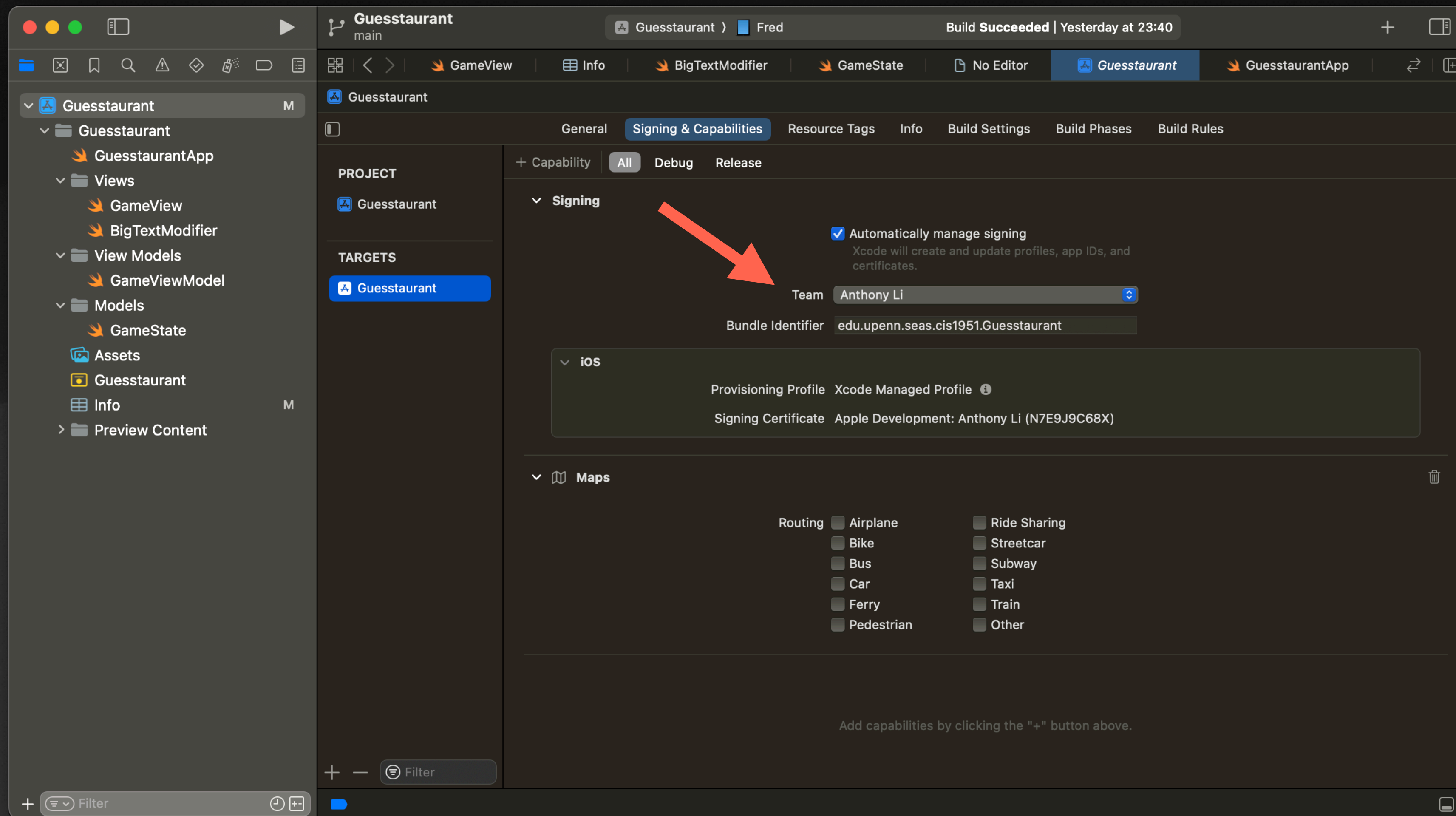
## Settings › Privacy & Security



# 3 Sign into your Apple ID in Xcode Settings



# 4 Set your Team under Signing & Capabilities



# 5 Select your device



# 6 Build and run!

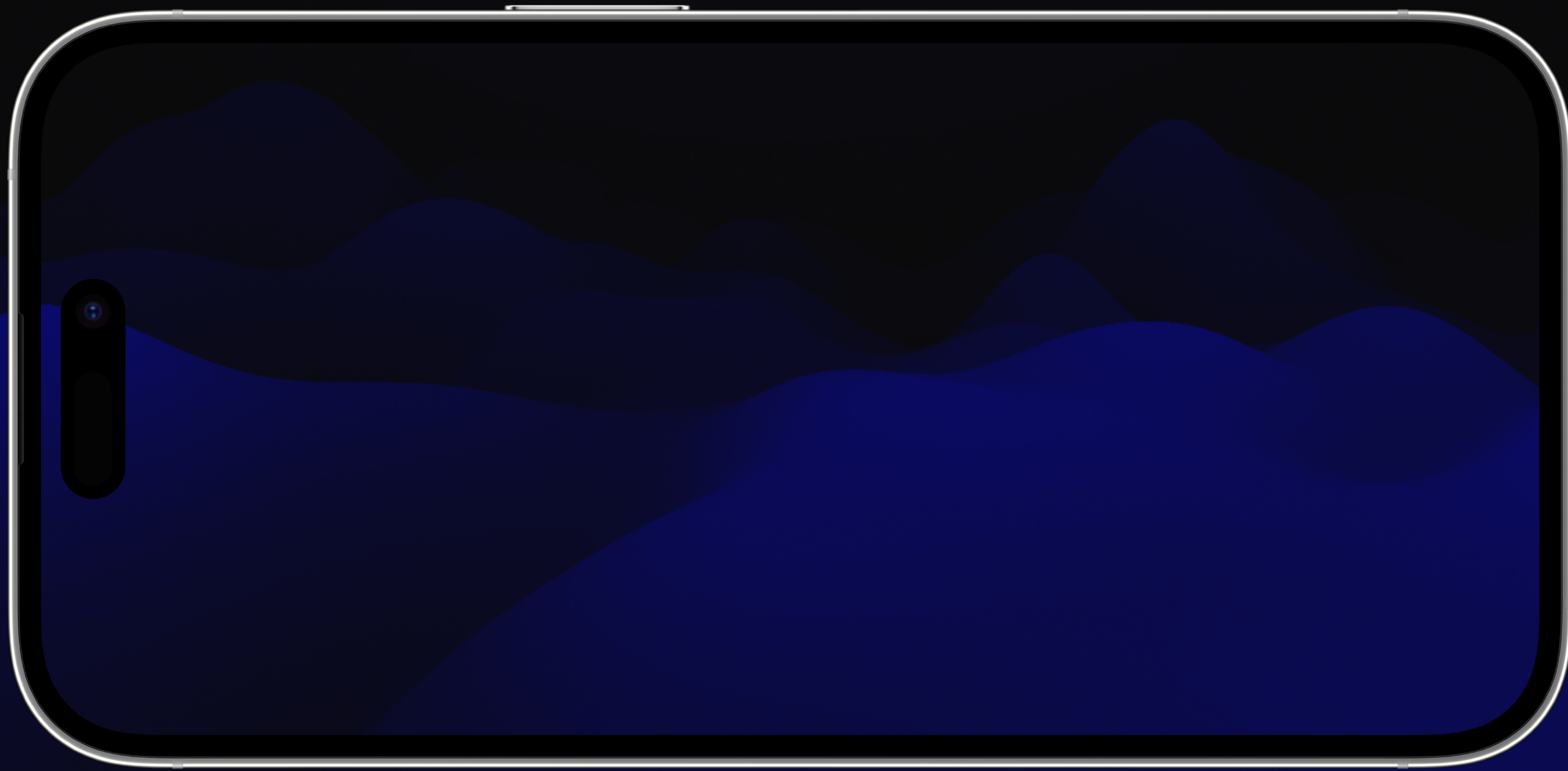


(DALL-E has mastered Swift for Aliens)

**Coding time!**

<https://github.com/cis1951/lec7-code>







# HW3

[title pending]

- Will be released **Monday, 3/11**
- Due **Monday, 3/25**
- [further details pending]

