ARKit 初探

从 Apple’s Template 到 ARGitHubCommits

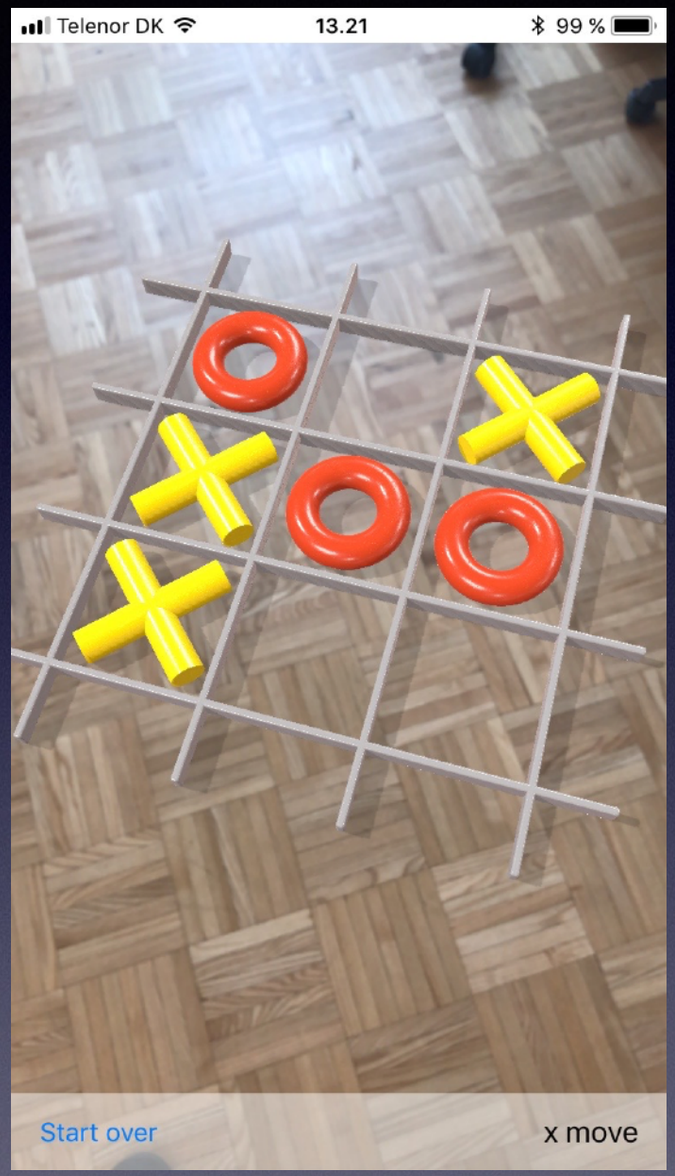
宋奎熹

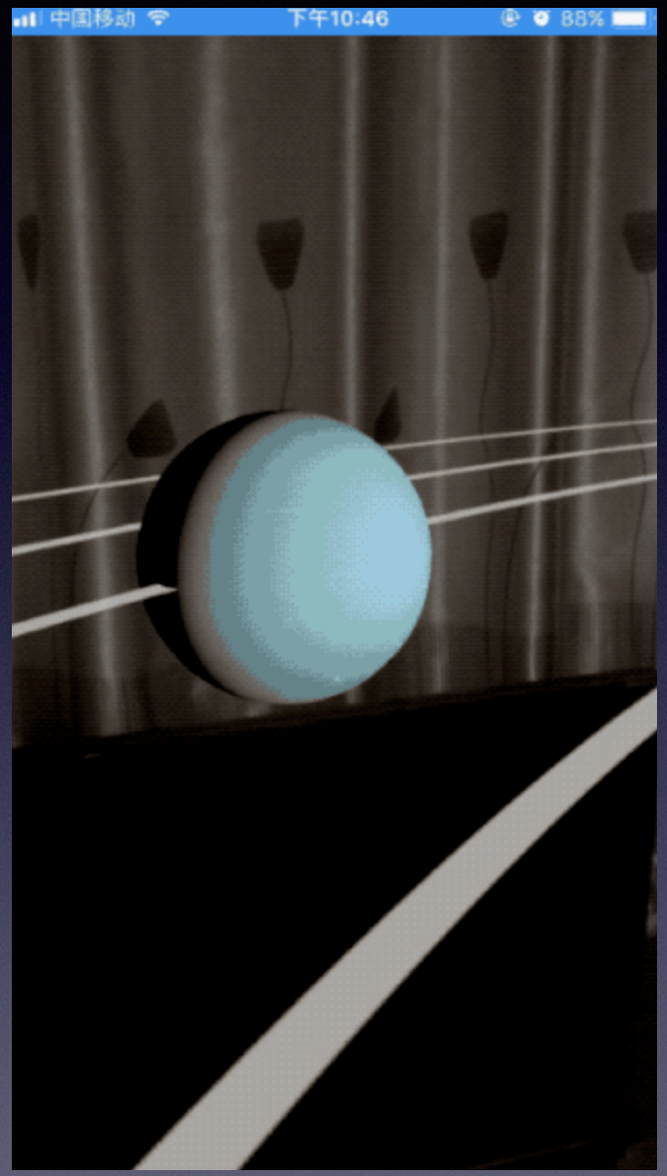
Outline

- ARKit Introduction
- Apple’s Template
- ARKit API
- ARGitHubCommits

ARKit Introduction

「iOS 11 引入了新的 ARKit 框架，让您轻松创建无可比拟的 iPhone 和 iPad 增强现实体验。通过将数字对象和信息与您周围的环境相融合，ARKit 为 App 解开了屏幕之缚，带领着它们跨越屏幕的界限，让它们以全新的方式与现实世界交流互动。」


Telenor DK
13.21
99 %
Start over
x move


中国移动
下午10:46
88%

Device Functions

- 视觉惯性里程计
- ARKit 使用**视觉惯性里程计 (VIO)** 以精准地追踪四周的环境。VIO 能**结合相机传感器数据与 CoreMotion 数据**，这两份数据让设备**无需额外的校准**，就能以**高精度来感测它在房间内的移动**。
- 场景理解和照明估计
- 使用 ARKit，iPhone 和 iPad 可以分析相机视图呈现的场景，并在房间中**找到水平面**。 ARKit 可以检测桌子和地板等水平面，并能跟踪及**放置物品在较小的特定点上**。 ARKit 还能利用相机传感器来**估计场景中可用光的总量**，并将正确的光亮度应用在虚拟对象上。
- 高性能硬件和渲染优化
- ARKit 可在 Apple **A9 及更高的处理器**上运行。 这些处理器提供了具有突破性的性能，可实现快速场景理解，并允许您在现实世界场景上构建出精细迷人的虚拟内容。 您还可以对 **Metal，SceneKit 以及 Unity 和 Unreal Engine (虚幻引擎)** 等第三方工具中的 ARKit 优化加以利用。

Requirement

- 软件
  - Xcode 9
  - iOS 11
  - macOS 10.12.4 及以上版本（为了支持 Xcode 9）
- 硬件
  - 处理器为 A9 及以上的 iPhone 或 iPad 设备（iPhone 6s 为 A9 处理器）

AR Work Progress

捕捉真实世界

建立虚拟世界

虚拟世界和
现实世界相结合

的同时

维持世界追踪

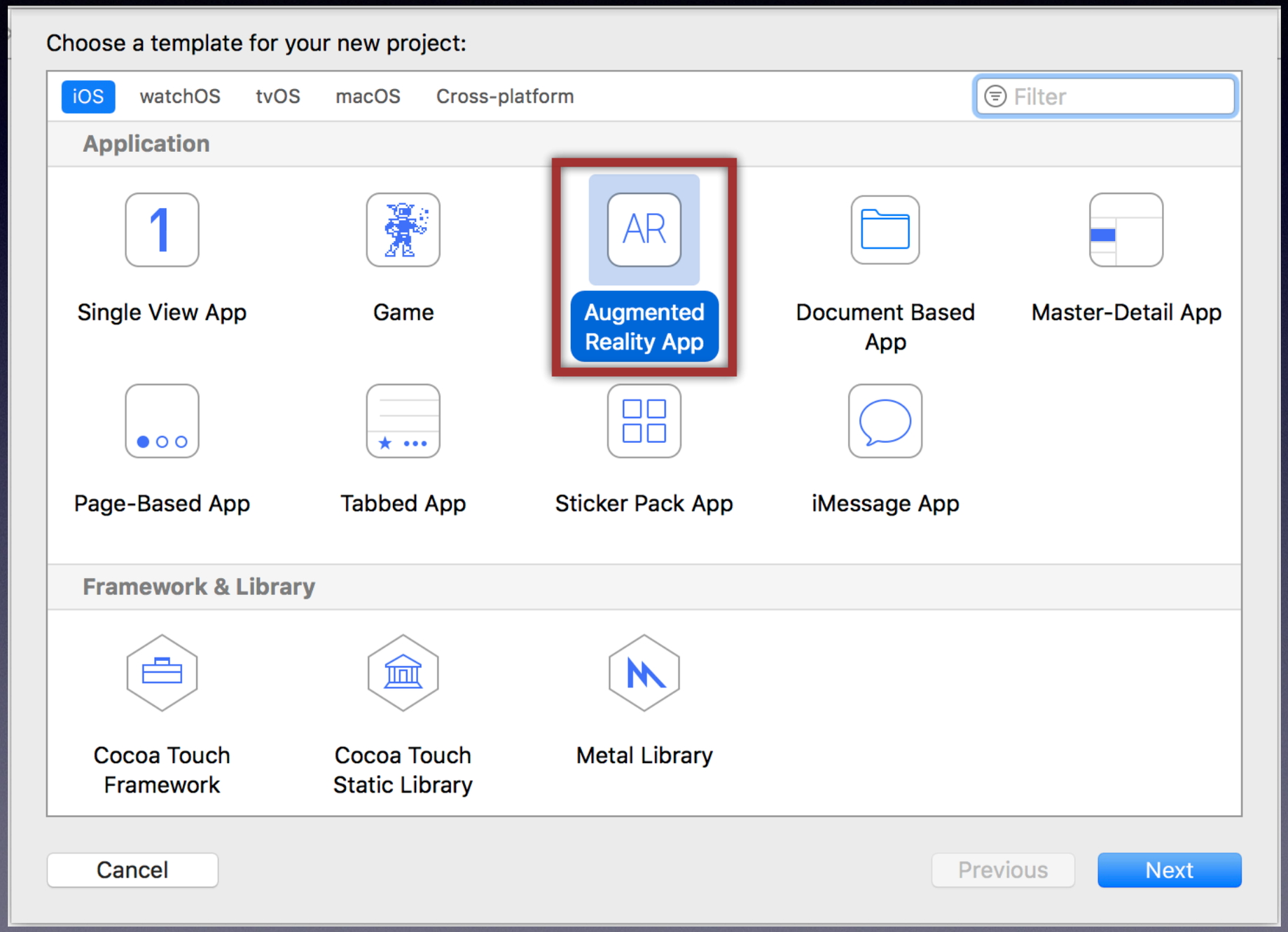
进行场景解析

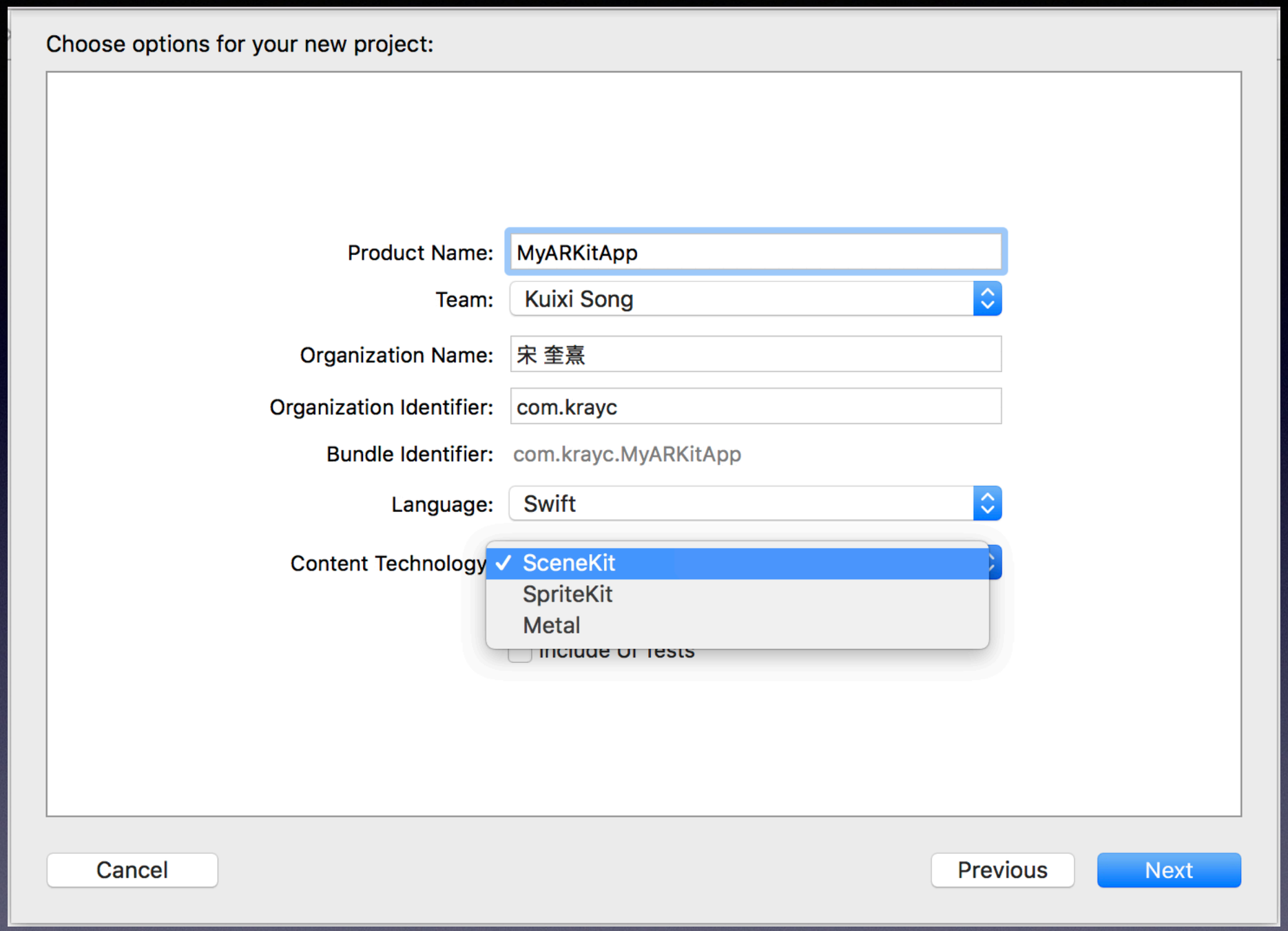
处理与
虚拟世界的互动

ARKit

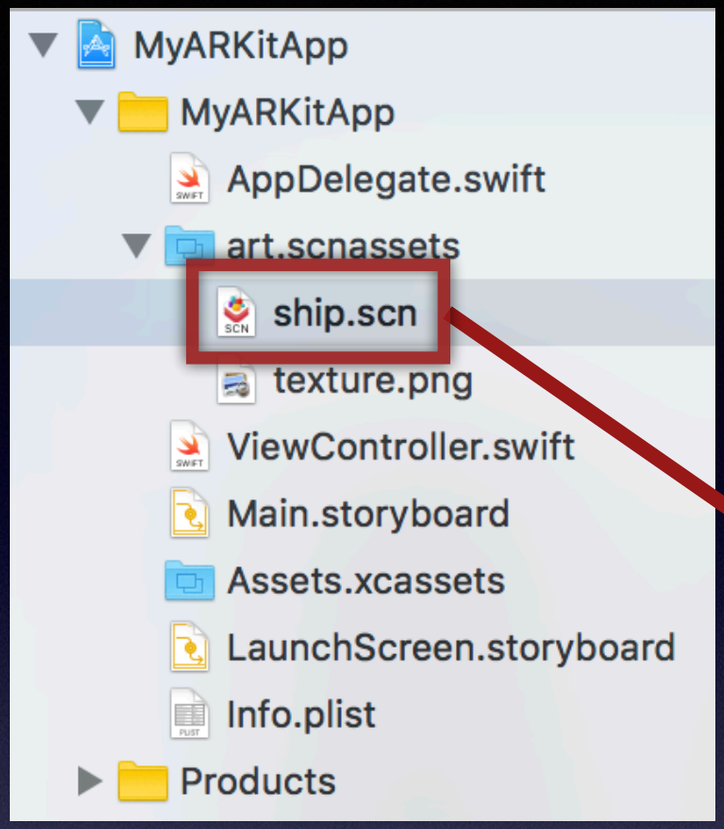
SceneKit

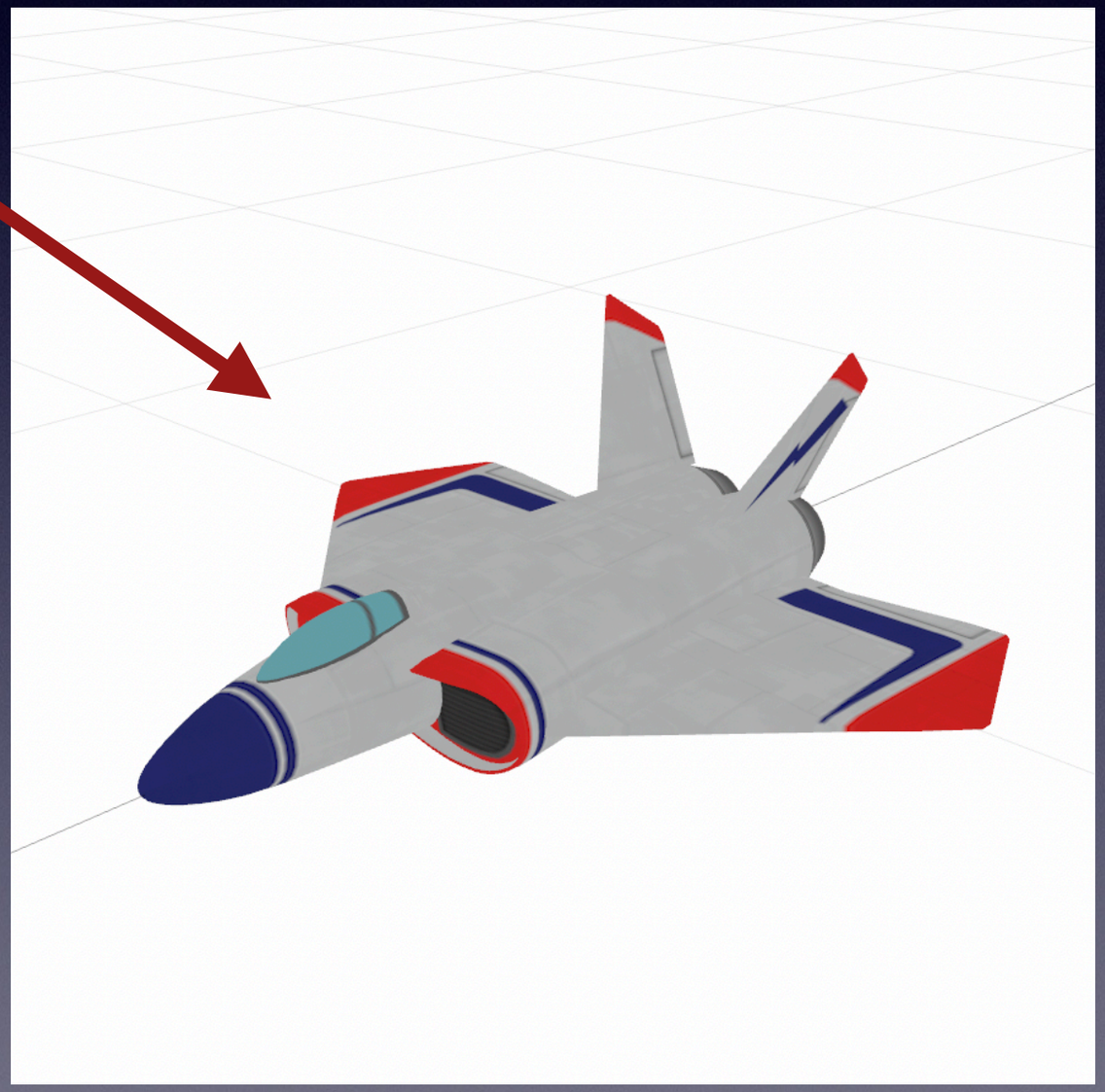
Apple's Template



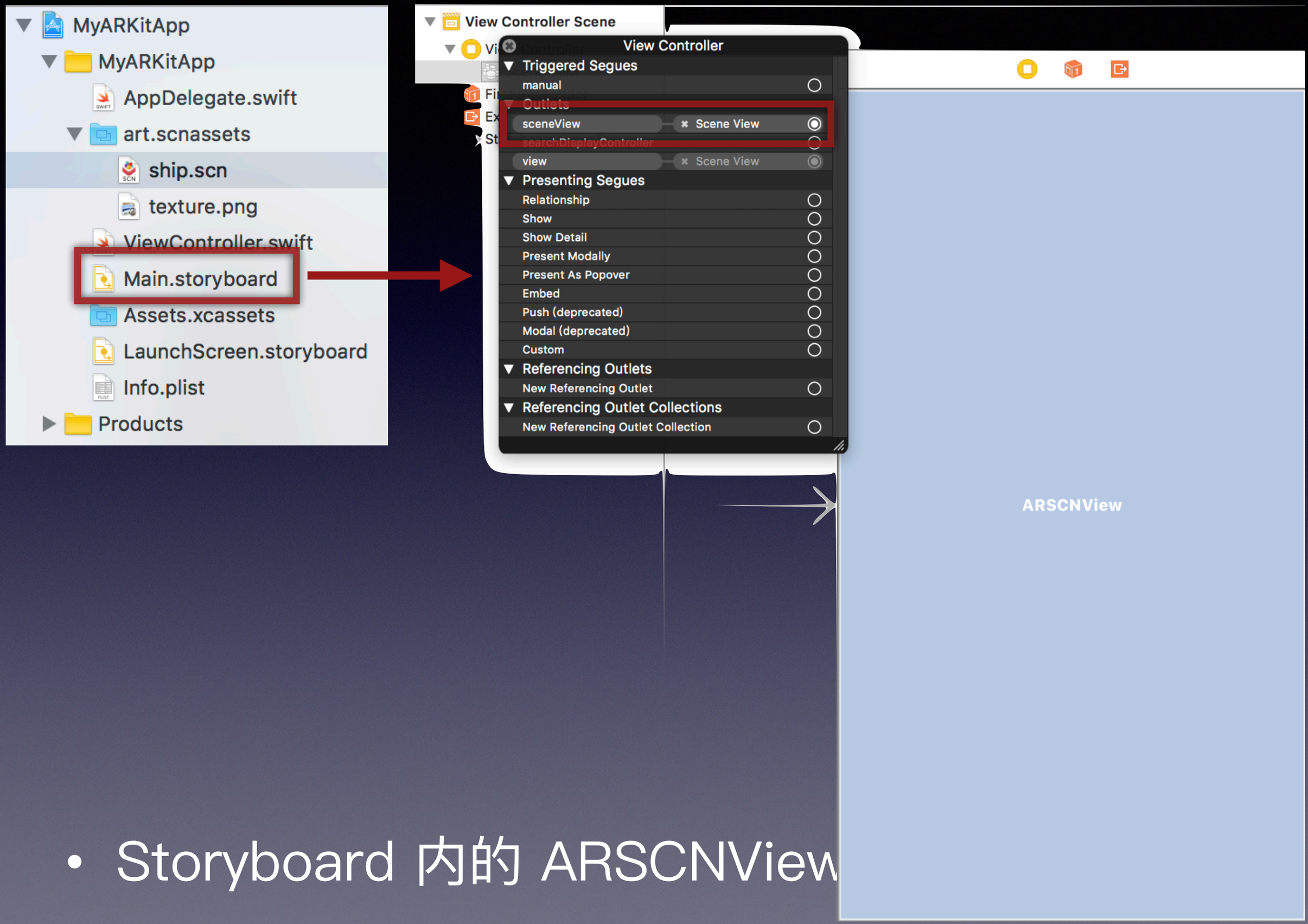


- 可选择 **SceneKit**、SpriteKit、Metal


MyARKitApp
MyARKitApp
AppDelegate.swift
art.scnassets
ship.scn
texture.png
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products



- 模板内自带 3D 模型



- Storyboard 內的 ARSCNView

```
class ViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!

    override func viewDidLoad() {

        super.viewDidLoad()

        // Set the view's delegate

        sceneView.delegate = self

        // Show statistics such as fps and timing information

        sceneView.showsStatistics = true

        // Create a new scene

        let scene = SCNScene(named: "art.scnassets/ship.scn")!

        // Set the scene to the view

        sceneView.scene = scene

}
```

建立虚拟世界

```
override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)
    // Create a session configuration  (6 DOF)
    let configuration = ARWorldTrackingConfiguration()
    // Run the view's session
    sceneView.session.run(configuration)
}
override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)
    // Pause the view's session
    sceneView.session.pause()
}
```

维持世界追踪

```
// Override to create and configure nodes for anchors added to the view's session.
    func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
        let node = SCNNode()
        return node
    }
    func session(_ session: ARSession, didFailWithError error: Error) {
// Present an error message to the user
    }
    func sessionWasInterrupted(_ session: ARSession) {
// Inform the user that the session has been interrupted, for example, by presenting an overlay
    }
    func sessionInterruptionEnded(_ session: ARSession) {
// Reset tracking and/or remove existing anchors if consistent tracking is required
    }
}
```

进行场景解析

渲染虚拟世界和现实世界

SCNView

继承

ARSCNView

ARSKView

场景

SCNScene

ARSession

会话

虚拟世界节点

现实世界信息

世界追踪配置

SCNNode

ARFrame

ARConfiguration

ARAnchor

ARCamera

现实世界锚点信息

摄像机信息

ARSCNViewDelegate

- func renderer(SCNSceneRenderer, nodeFor: ARAnchor)
- func renderer(SCNSceneRenderer, didAdd: SCNNode, for: ARAnchor)
- func renderer(SCNSceneRenderer, willUpdate: SCNNode, for: ARAnchor)
- func renderer(SCNSceneRenderer, didUpdate: SCNNode, for: ARAnchor)
- func renderer(SCNSceneRenderer, didRemove: SCNNode, for: ARAnchor)

ARSessionDelegate

- func session(ARSession, didUpdate: ARFrame)
- func session(ARSession, didAdd: [ARAnchor])
- func session(ARSession, didUpdate: [ARAnchor])
- func session(ARSession, didRemove: [ARAnchor])

如果使用了 ARSCNViewDelegate 或 ARSKViewDelegate，那上述几个灰色方法不必实现

How to add objects?

- For example, by…

Auto detection

Tapping on screen

More ways…

Auto detection

- Using ARSCNViewDelegate method

```
func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {

        if let planeAnchor = anchor as? ARPlaneAnchor {

            let node = SCNNode()

            node.geometry = SCNBox(width: CGFloat(planeAnchor.extent.x), height: CGFloat(planeAnchor.extent.y), length: CGFloat(planeAnchor.extent.z), chamferRadius: 0)

            return node

        }

        return nil

    }
```

Tapping on screen

- Using UITapGestureRecognizer

```
@objc func didTap(_ sender: UITapGestureRecognizer) {
        let location = sender.location(in: sceneView)
        let hitResults = sceneView.hitTest(location, types: .featurePoint)
    if let result = hitResults.first {
        let box = SCNBox(width: 0.1, height: 0.1, length: 0.1, chamferRadius: 0)
        let boxNode = SCNNode(geometry: box)
        boxNode.position = SCNVector3(x: result.worldTransform.columns.3.x,
                                      y: result.worldTransform.columns.3.y,
                                      z: result.worldTransform.columns.3.z)
        sceneView.scene.rootNode.addChildNode(boxNode)
    }
}
```

ARHitTestResult

- featurePoint
  - 返回当前图像中 Hit-testing 射线经过的 3D 特征点
- estimatedHorizontalPlane
  - 返回当前图像中 Hit-testing 射线经过的预估平面。
- existingPlaneUsingExtent
  - 返回当前图像中 Hit-testing 射线经过的**有大小范围的平面**。
- existingPlane
  - 返回当前图像中 Hit-testing 射线经过的**无限大小的平面**。

Update object’s position

- Using ARSessionDelegate method

```
func session(_ session: ARSession, didUpdate frame: ARFrame) {

    if boxNode != nil {

        let mat = frame.camera.transform.columns.3

        boxNode?.position = SCNVector3Make(

            (mat.x) * 3, (mat.y) * 3, (mat.z) * 3 – 0.5

        )

    }

}
```

ARGitHubCommits

1. 获取 GitHub 的 Commits 数据
2. 建立 ARSCNView，开始 ARSession
3. **提示用户移动手机**，探测水平面

```
let configuration = ARWorldTrackingConfiguration()

configuration.planeDetection = .horizontal
```

4. 探测成功后，在 ARSCNView 中的 SCNScene 中加入各个 SCNNode

Talk is cheap.

Show me the code.

项目地址：https://github.com/songkuixi/ARGitHubCommits

Reference

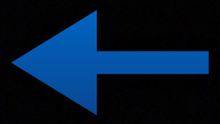
- https://developer.apple.com/cn/arkit/
- https://developer.apple.com/documentation/arkit
- http://blog.csdn.net/u013263917/article/details/72903174
- https://mp.weixin.qq.com/s/DxPHo6j6pJQuhdXM_5K4qw
- https://github.com/olucurious/Awesome-ARKit
- https://github.com/songkuixi/ARGitHubCommits

谢谢!

WeChat
@krayc425

GitHub
@songkuixi

Weibo
@滑滑鸡



ARSCNView

- 将**摄像机**捕捉到的真实世界的视频作为背景。
- 处理**光照估计信息**，不断更新画面的光照强度。
- **将 SCNNode 与 ARAnchor 绑定**，也就是说当添加一个 SCNNode 时，ARSCNView 会同时添加一个 ARAnchor 到 ARKit 中。
- 将 SceneKit 中的坐标系结合到 AR-World 的坐标系中，不断**渲染 SceneKit 场景到真实世界的画面中。**

World Tracking

- 由 ARSession 类负责，输出是 ARFrame。ARFrame 中包含有渲染虚拟世界所需的所有信息（Anchors 和 Camera）。
- 追踪设备的**位置**以及**旋转**，这里的两个信息均是相对于设备起始时的信息。
- 追踪物理距离(以“米”为单位)，例如 ARKit 检测到一个平面，我们希望知道这个平面有多大。
- 追踪我们手动添加的希望追踪的点，例如我们手动添加的一个虚拟物体。

Tracking Quality

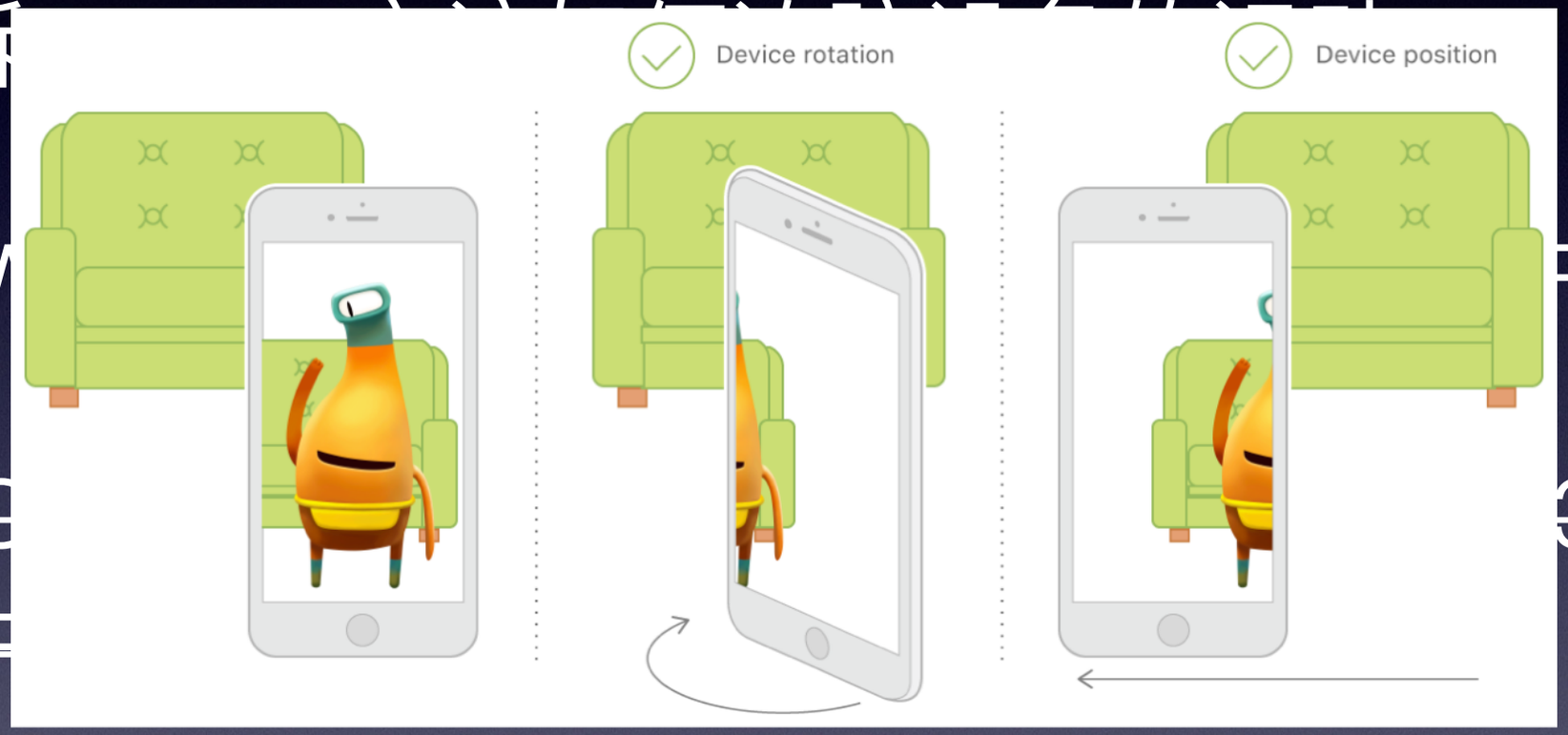
- 运动传感器**不能停止工作**。如果运动传感器停止了工作，那么就无法拿到设备的运动信息。根据我们之前提到的世界追踪的工作原理，毫无疑问，追踪质量会下降甚至无法工作。

- 真实世界的场景需要**有一定特征点可追踪**。世界追踪需要不断分析和追踪捕捉到的图像序列中特征点，如果图像是一面白墙，那么特征点非常少，那么追踪质量就会下降。

- 设备移动**速度不能过快**。如果设备移动太快，那么 ARKit 无法分析出不同图像帧之中的特征点的对应关系，也会导致追踪质量下降。

ARConfiguration

- 告诉
- ARV F)
- ~~ARC~~ 3 ~~DOF~~



- ARFaceTrackingConfiguration（iPhone X Only）

ARAnchor

- 可以把 ARAnchor（锚点）理解为真实世界中的某个点或平面，anchor 中包含位置信息和旋转信息。拿到 anchor 后，可以在该 anchor 处放置一些虚拟物体
- 与 SCNNode 可以绑定
- 子类：ARPlaneAnchor