Sample Code

Tracking and visualizing planes

Detect surfaces in the physical environment and visualize their shape and location in 3D space.

Download

iOS 15.6+ | iPadOS 15.6+ | Xcode 16.0+

Overview

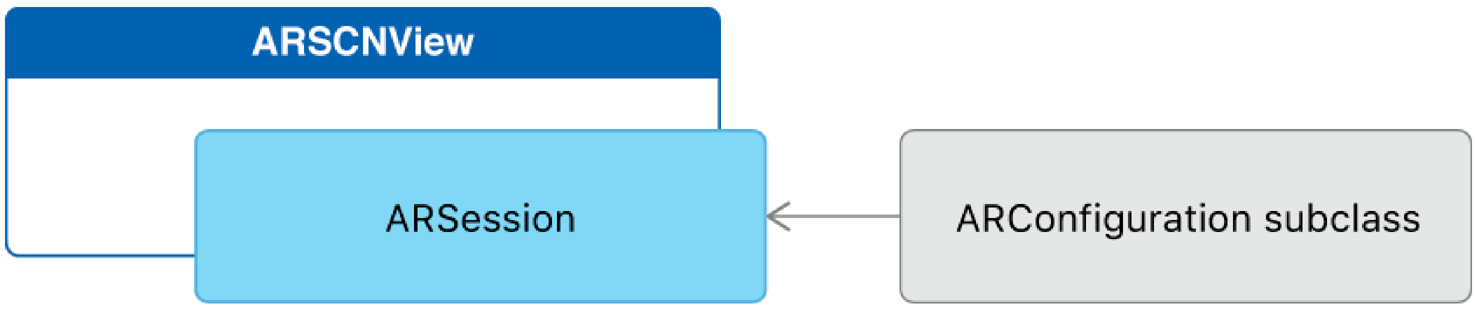
This sample app runs an ARKit world tracking session with content displayed in a SceneKit view. To demonstrate plane detection, the app visualizes both the estimated shape of and a bounding rectangle for each detected `ARPlaneAnchor` object. On supported devices, ARKit can recognize many types of real-world surfaces, so the app also labels each detected plane with identifying text.

Getting started

ARKit requires iOS 11 and a device with an A9 (or later) processor. ARKit is not available in iOS Simulator. Building the sample code requires Xcode 9 or later.

Configure and run the AR session

The `ARSCNView` class is a SceneKit view that includes an `ARSession` object that manages the motion tracking and image processing required to create an augmented reality (AR) experience. However, to run a session you must provide a session configuration.



The ARWorldTrackingConfiguration class provides high-precision motion tracking and enables features to help you place virtual content in relation to real-world surfaces. To start an AR session, create a session configuration object with the options you want (such as plane detection), then call the run(_:options:) method on the session object of your ARSCNView instance:

```
let configuration = ARWorldTrackingConfiguration()
configuration.planeDetection = [.horizontal, .vertical]
sceneView.session.run(configuration)
```

Run your session only when the view that will display it is onscreen.

Important

If your app requires ARKit for its core functionality, use the arkit key in the UIRequiredDeviceCapabilities section of your app's Info.plist file to make your app available only on devices that support ARKit. If AR is a secondary feature of your app, use the isSupported property to determine whether to offer AR-based features.

Place 3D content for detected planes

After you've set up your AR session, you can use SceneKit to place virtual content in the view.

When plane detection is enabled, ARKit adds and updates anchors for each detected plane. By default, the ARSCNView class adds an SCNNode object to the SceneKit scene for each anchor. Your view's delegate can implement the renderer(_:didAdd:for:) method to add content to the scene. When you add content as a child of the node corresponding to the anchor, the ARSCNView class automatically moves that content as ARKit refines its estimate of the plane's position.

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnch
    // Place content only for anchors found by plane detection.
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }
    // Create a custom object to visualize the plane geometry and extent.
    let plane = Plane(anchor: planeAnchor, in: sceneView)

    // Add the visualization to the ARKit-managed node so that it tracks
    // changes in the plane anchor as plane estimation continues.
    node.addChildNode(plane)
```

ARKit offers two ways to track the area of an estimated plane. A plane anchor's `geometry` describes a convex polygon tightly enclosing all points that ARKit currently estimates to be part of the same plane (easily visualized using `ARSCNPlaneGeometry`. ARKit also provides a simpler estimate in a plane anchor's `extent` and `center`], which together describe a rectangular boundary (easily visualized using `SCNPlane`.

```
// Create a mesh to visualize the estimated shape of the plane.
guard let meshGeometry = ARSCNPlaneGeometry(device: sceneView.device!)
    else { fatalError("Can't create plane geometry") }
meshGeometry.update(from: anchor.geometry)
meshNode = SCNNode(geometry: meshGeometry)

// Create a node to visualize the plane's bounding rectangle.
let extentPlane: SCNPlane = SCNPlane(width: CGFloat(anchor.extent.x), height: CGFloa
extentNode = SCNNode(geometry: extentPlane)
extentNode.simdPosition = anchor.center

// `SCNPlane` is vertically oriented in its local coordinate space, so
// rotate it to match the orientation of `ARPlaneAnchor`.
extentNode.eulerAngles.x = -.pi / 2
```

ARKit continually updates its estimates of each detected plane's shape and extent. To show the current estimated shape for each plane, this sample app also implements the `renderer(_:didUpdate:for:)` method, updating the `ARSCNPlaneGeometry` and `SCNPlane` objects to reflect the latest information from ARKit.

```
func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: AR
    // Update only anchors and nodes set up by `renderer(_:didAdd:for:)`.
    guard let planeAnchor = anchor as? ARPlaneAnchor,
        let plane = node.childNodes.first as? Plane
        else { return }

    // Update ARSCNPlaneGeometry to the anchor's new estimated shape.
    if let planeGeometry = plane.meshNode.geometry as? ARSCNPlaneGeometry {
        planeGeometry.update(from: planeAnchor.geometry)
    }

    // Update extent visualization to the anchor's new bounding rectangle.
    if let extentGeometry = plane.extentNode.geometry as? SCNPlane {
        extentGeometry.width = CGFloat(planeAnchor.extent.x)
        extentGeometry.height = CGFloat(planeAnchor.extent.z)
        plane.extentNode.simdPosition = planeAnchor.center
    }

    // Update the plane's classification and the text position
    if #available(iOS 12.0, *),
        let classificationNode = plane.classificationNode,
        let classificationGeometry = classificationNode.geometry as? SCNText {
        let currentClassification = planeAnchor.classification.description
        if let oldClassification = classificationGeometry.string as? String, oldClas
            classificationGeometry.string = currentClassification
            classificationNode.centerAlign()
        }
    }

}
```

On some hardware, ARKit can also classify detected planes, reporting which kind of common real-world surface that plane represents (for example, a table, floor, or wall). In this example, the `renderer(_:didUpdate:for:)` method also displays and updates a text label to show that information when running on hardware that supports it.

```
// Display the plane's classification, if supported on the device
if #available(iOS 12.0, *), ARPlaneAnchor.isClassificationSupported {
    let classification = anchor.classification.description
    let textNode = self.makeTextNode(classification)
    classificationNode = textNode
    // Change the pivot of the text node to its center
    textNode.centerAlign()
    // Add the classification node as a child node so that it displays the classifi
    extentNode.addChildNode(textNode)
}
```

See Also

Surface Detection

`class ARPlaneAnchor`

An anchor for a 2D planar surface that ARKit detects in the physical environment.

`class ARMeshAnchor`

An anchor for a physical object that ARKit detects and recreates virtually using a polygonal mesh.