RealityKit / SpatialTrackingSession

Class

SpatialTrackingSession

An object that incorporates spatial tracking capabilities into your RealityKit apps.

iOS 18.0+ | iPadOS 18.0+ | visionOS 2.0+

```
final class SpatialTrackingSession
```

Overview

`SpatialTrackingSession` helps you manage spatial tracking capabilities, such as world and hand tracking, in your RealityKit app. You can decide which AR data you need and how you want to use them by running a corresponding `SpatialTrackingSession.Configuration`.

You can specify the following spatial tracking capabilities in a configuration:

- The anchoring targets you want RealityKit to track, such as planes, images, and hands.
- In iOS, the scene-understanding data that system behaviors have access to, such as shadows, occlusion, and physics.
- In iOS, the camera feed (front or back) that `spatialTracking` uses.

The following is an example of how to enable the camera, implement plane anchoring, use scene-understanding data for shadow and occlusion, and use the back camera in iOS:

```
let configuration = SpatialTrackingSession.Configuration(
    tracking: [.camera, .plane],
    sceneUnderstanding: [.shadow, .occlusion],
    camera: .back)
let session = SpatialTrackingSession()
await session.run(configuration)
```

SpatialTrackingSession configures ARKit for you based on the configuration you pass in. It also checks whether a device supports certain requested capabilities, and requests user authorizations if needed.

As an asynchronous function call, run(_:) returns an optional SpatialTrackingSession.UnavailableCapabilities, which contains both unsupported capabilities and unauthorized capabilities. This can help you to handle errors gracefully, such as informing someone to turn on the relevant permissions or reduce the feature set. Here's an example in iOS:

```swift
let configuration = SpatialTrackingSession.Configuration(
    tracking: [.camera, .plane],
    sceneUnderstanding: [.shadow, .occlusion],
    camera: .back)
let session = SpatialTrackingSession()
if let unavailableCapabilities = await session.run(configuration) {
    if unavailableCapabilities.missingCameraAuthorization {
        print("Camera access requires user permission.")
        return
    }
    if unavailableCapabilities.anchor.contains(.plane) {
        print("The device doesn't support plane tracking.")
    }
    if !unavailableCapabilities.sceneUnderstanding.isEmpty {
        print("The device doesn't support scene understanding.")
    }
}
```

After running SpatialTrackingSession, RealityKit apps start using the AR data in the following ways:

- Anchor tracking starts to work for the chosen capabilities as follows:
  - In visionOS, you can access the transform information of the entities with AnchoringComponent.
  - In iOS, entities with AnchoringComponent start to get the real-world anchors to anchor onto.
- System behavior starts to use scene-understanding data. In the example above, virtual contents start to cast shadows onto real-world meshes, and experience occlusion from those meshes.
- Setting camera to .spatialTracking starts to show the configured camera feed as a RealityView background.

To modify the configuration, run `SpatialTrackingSession` again with the new configuration. To stop `SpatialTrackingSession`, you can call `stop()`.

> Note
>
> RealityKit stops `SpatialTrackingSession` automatically when it goes out of scope and deinitializes.

Other methods for incorporating AR content in your apps

Although `SpatialTrackingSession` offers you an easy way to configure AR capabilities in your app, there are other options available.

Use the default spatial tracking session

In iOS, if you don't set up `SpatialTrackingSession` manually, RealityKit runs the default session with the following configuration:

```
SpatialTrackingSession.Configuration(
    tracking: [.camera, .world, .plane, .object, .image, .face],
    sceneUnderstanding: [.occlusion, .shadow, .collision, .physics],
    camera: .back)
```

Use an anchor entity alone in visionOS

In visionOS, if you don't set up a spatial tracking session manually, the system anchors your `AnchorEntity` in a privacy-preserving manner. In other words, the app doesn't get read access to any transform information from the `AnchorEntity`.

The app requests permission to read transform information when you set up a spatial tracking session.

> Important
>
> In visionOS, avoid applying a transform that a spatial tracking session provides to an `AnchorEntity` instance to an entity because it makes the entity lag behind the anchor by one frame.

You can avoid the one frame lag when rendering an entity with the transform of an `AnchorEntity` by adding it as a descendent of that anchor entity. Alternatively, you can add a `ModelComponent` instance that contains the relevant content for rendering directly to the anchor entity.

Tip

Only apply the transform from a spatial tracking session instance to non-rendering purposes, such as collision detection or custom gestures.

Use an ARKit session for tracking

You can set up an ARKit session manually and receive an anchor transform directly from ARKit by subscribing to the ARKit updates. You can then set the transform of your entities manually. This gives you full access to the anchor properties, but requires more setup on your part. With `SpatialTrackingSession`, RealityKit handles that for you and maintains your `AnchorEntity` alignment to the target anchor. In iOS, you manage and run the ARKit session if you run `SpatialTrackingSession` with `run(_:session:arConfiguration:)`.

Topics

Structures

`struct Configuration`

A type for configuring the spatial tracking session.

`struct UnavailableCapabilities`

A type that contains the unavailable capabilities of the current spatial tracking session.

Initializers

`init()`

Creates a spatial tracking session with default settings.

Instance Methods

`func run(SpatialTrackingSession.Configuration) async -> SpatialTrackingSession.UnavailableCapabilities?`

Runs the spatial tracking session with the specified configuration.

`func run(SpatialTrackingSession.Configuration, session: ARSession, arConfiguration: ARConfiguration) async -> SpatialTrackingSession.UnavailableCapabilities?`

Runs the spatial tracking session with a spatial tracking configuration, an AR session, and an AR configuration.

`func stop() async`

Stops the current spatial tracking session.

Relationships

Conforms To

`Sendable`, `SendableMetatype`

See Also

Spatial tracking

`struct Configuration`

A type for configuring the spatial tracking session.

`struct AnchorCapability`

A type that defines various anchor tracking capabilities.

`struct SceneUnderstandingCapability`

Defines how system behaviors use scene understanding.

`enum Camera`

Defines the camera feed the RealityView renders.

`struct UnavailableCapabilities`

A type that contains the unavailable capabilities of the current spatial tracking session.