Article

Discovering and tracking spatial game controllers and styli

Receive controller and stylus input to interact with content in your augmented reality app.

Overview

The Game Controller framework provides the ability to discover spatial game controllers and stylus, allows you to connect, read button or thumbstick inputs, and play haptics. After you connect to a device, you use RealityKit or ARKit to combine tracking data with input from the device.

Configure your project

To begin developing with spatial game controllers, you need to configure your Xcode project. To add the spatial game controller profile to your project, perform the following steps:

1. In Xcode, select your project in Xcode's project navigator.
2. Select your project's target.
3. Click the Signing & Capabilities tab in the project editor.
4. Add the Game Controller capability.
5. Select the Spatial Gamepad profile.

> **Note**
>
> You don't need to enable the Spatial Gamepad profile if your app only supports stylus input.

Discover a controller or stylus

The system can notify your app when a spatial game controller connects or disconnects by listening for `GCControllerDidConnect` and `GCControllerDidDisconnect`. A notification that includes information as to whether the controller provides spatial input:

```
NotificationCenter.default.addObserver(
        forName: NSNotification.Name.GCControllerDidConnect,
        object: nil,
        queue: nil) { notification in
    if let controller = notification.object as? GCController {
        switch controller.productCategory {
            case GCProductCategorySpatialController:
                // A spatial controller connected.
            default:
                // A standard controller connected.
        }
    }
}
```

More than one controller can connect to a device at a time. You can use the connection notification to track each connection as they happen, or check `controllers()` to iterate through an up-to-date list of the currently connected controllers.

To get notifications for styli, use `GCStylusDidConnectNotification` and `GCStylusDidDisconnectNotification`. These notifications provide a `GCStylus`, and you can get a list of all currently connected styli by querying `styli`.

> **Note**
>
> Use `GCControllerDidConnect` and `GCStylusDidConnectNotification` when your app launches to get the initial connection state. Checking for controllers and styli isn't synchronous and may return an empty list even with an accessory in a connected state.

Handle input mapping

You use `input` to access the button and thumbstick inputs of a spatial controller. When you work with spatial game controllers, the input button mapping expose the following inputs:

```
input.buttons[.a] // Cross button (Right), Square button (Left)
input.buttons[.b] // Circle button (Right), Triangle button (Left)
input.buttons[.gripButton] // Grip button
input.buttons[.trigger] // Trigger button
input.buttons[.thumbstickButton] // Thumbstick "press"
input.buttons[.menu] // Menu button
input.dpads[.thumbstick] // Joystick
```

Use `input` to access inputs from a spatial stylus accessory. A stylus exposes the following inputs:

```
input.buttons[.stylusTip] // Tip pressure sensor
input.buttons[.stylusPrimaryButton] // Primary side button
input.buttons[.stylusSecondaryButton] // Secondary side button
```

For information on polling for input and receiving callbacks, see Handling input events. For more information on how to play haptics, see Playing Haptics on Game Controllers.

Track spatial position with RealityKit anchor entities

In RealityKit, an `AnchorEntity` provides a way to tether virtual content to physical locations or objects in your real work space. For example, an image in your environment, your hands, or a spatial game controller. On visionOS, accessory anchoring works in immersive and shared spaces.

Use `AnchoringComponent.AccessoryAnchoringSource` with a `GCController` or `GCStylus` to anchor virtual content onto the accessory. Each controller and stylus accessory has a list of possible locations you can anchor to, and depends on the accessory you use. You can anchor virtual content to a location on the accessory by specifying a `name` from a list of possible `accessoryLocations`.

```
let device = // A connected controller or stylus.
guard let source = try await AnchoringComponent.AccessoryAnchoringSource(device: dev
    // Get a list of location names available for the device.
    let names = source.accessoryLocationNames

    // Get a named location on a controller, like `aim`, `grip`, or `grip_surface`.
    let aimLocation = source.locationName(named: "aim")

    // Create an entity that targets a location you want to anchor to.
    let aimEntity = AnchorEntity(.accessory(from: source, location: aimLocation),
                                 trackingMode: .predicted)
```

```
}
```

For apps that don't depend on high location accuracy, use the `predicted` tracking mode. If you need higher location accuracy — at the cost of higher latency — use `continuous` tracking mode.

Before using `SpatialTrackingSession` to get the transforms of a spatial game controller, your app needs request permission to track an accessory. Set `NSAccessoryTrackingUsageDescription` in your app's `Info.plist` file that explains how your app intends to use tracking information.

```
// Configure a spatial tracking session.
let configuration = SpatialTrackingSession.Configuration(tracking: [.accessory])
let session = SpatialTrackingSession()
await session.run(configuration)

// Get the anchor transform from an entity.
let aimTransform = aimEntity.transformMatrix(relativeTo: nil)
```

If you use ARKit, tracking works similarly to the object and image tracking APIs. For more information about tracking accessories, see Tracking accessories in volumetric windows.

See Also

Game controllers

{} Supporting Game Controllers

Support a physical controller or add a virtual controller to enhance how people interact with your game through haptics, lighting, and motion sensing.

Letting players use their second-generation Siri Remote as a game controller

Support the second-generation Siri Remote as a game controller in your Apple TV game.

`protocol` GCDevice

A protocol that defines a common interface for game input devices.

`class` GCController

A representation of a real game controller, a virtual controller, or a snapshot of a controller.

`class` GCRacingWheel

An object that represents a physical racing wheel controller connected to a device.

`class` GCKeyboard

An object that represents a physical keyboard connected to a device.

`class` GCMouse

An object that represents a physical mouse connected to a device.

`class` GCStylus

An object that represents a physical stylus connected to the device.