Framework

Game Controller

Support hardware game controllers in your game.

iOS 7.0+ | iPadOS 7.0+ | Mac Catalyst 13.1+ | macOS 10.9+ | tvOS 9.0+ | visionOS 1.0+

Overview

Use Game Controller to support users interacting with your app using a physical or virtual game controller. Game controllers include third-party products, such as the DualShock 4, DualSense, and Xbox, as well as the mouse, keyboard, Siri Remote, and racing wheels.

To support game controllers, add the Game Controller capability (the `GCSupportsControllerUserInteraction` property) to your project, and Xcode adds the Game Controller framework automatically. Then, choose the types of controllers your app supports (the `GCSupportedGameControllers` property) under the Game Controllers capability on the Signing & Capabilities pane.

For controllers other than the racing wheel, follow these steps to process the game controller input in your app:

- To get a physical game controller object, register for specific notifications when users connect and disconnect game controllers. Alternatively, you can display a virtual controller for the user to interact with.
- Then get a profile object from the physical or virtual controller to access its input elements, such as buttons, triggers, thumbsticks, and directional pads. Profiles encapsulate the hardware details and layout of the input elements on the controller from your app.
- To process the input, either get the values directly from the elements or register callbacks for when the user changes their values. Apps running in visionOS receive input events only when the person is looking at the app's window.
- For controllers that support haptics, you can provide feedback to the user by creating an engine that manipulates the controller's actuators.

Users may remap game controller elements in Settings and Preferences, so be sure to display the correct input element in your interface. If the `hasRemappedElements` property is `true`, the user remapped elements and you can get the mapping between the actual and alias elements using the `mappedElementAlias(forPhysicalInputName:)` and `mappedPhysicalInputNames(forElementAlias:)` methods.

To support racing wheel devices in your macOS app, see Racing wheel device support.

Topics

Essentials

Game Controller updates

Learn about important changes to Game Controller.

`GCSupportsControllerUserInteraction`

A Boolean value indicating whether the app supports a game controller.

`GCSupportedGameControllers`

The types of game controller profiles that the app supports or requires.

`GCSupportsMultipleMicroGamepads`

A Boolean value indicating whether the physical Apple TV Remote and the Apple TV Remote app operate as separate game controllers.

Handling input events

Receive controller input using either polling or callbacks.

View controller

`class GCEventViewController`

A view controller that delivers input either from the responder chain to views, or from game controllers to profiles.

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.

Discovering and tracking spatial game controllers and styli

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

`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.

Game controller profiles

Input

Receive controller input in the way that best integrates with the flow of your game or game engine.

`class GCMotion`

A controller profile that supports orientation and motion.

`class GCDeviceBattery`

The charge level and state of a device’s battery.

`class GCDeviceHaptics`

The locations of haptic actuators on a game controller.

`class GCDeviceLight`

The colored light on a device.

Touch controller

Adding touch controls to games that support game controllers in iOS

Use touch input and virtual controllers to make your game available to players without controllers.

`class GCVirtualController`

A software emulation of a real controller that you configure specifically for your game.

Button elements and names

`protocol GCTouchedStateInput`

The common properties for an element that has touch state input.

`protocol GCPressedStateInput`

The common properties for an element that has press state input, such as input from a button.

Racing wheels

Racing wheel device support

Add support for racing wheel devices in macOS.

Game Controller framework migration from IOKit

Deploy an app that takes advantage of advanced game controller features using the Game Controller framework, while supporting game controllers on older macOS releases using IOKit.

Understanding game controller backward compatibility

Learn how macOS brings support for the latest game controllers to software that predates the introduction of the Game Controller framework.

`var kIOHIDGCSyntheticDeviceKey: String`

A key that specifies whether the device is a game controller synthetic HID device.

Aliases for backward compatibility

`typealias GCDeviceElement`

An alias for a symbol name for backward compatibility with a previous SDK version.

`typealias GCDeviceAxisInput`

An alias for a symbol name for backward compatibility with a previous SDK version.

`typealias GCDeviceButtonInput`

An alias for a symbol name for backward compatibility with a previous SDK version.

`typealias GCDeviceTouchpad`

An alias for a symbol name for backward compatibility with a previous SDK version.

`typealias GCDeviceDirectionPad`

An alias for a symbol name for backward compatibility with a previous SDK version.

Deprecated symbols

Deprecated symbols

Protocols

`protocol GCPhysicalInputExtents`

Physical extents scale the normalized value reported by `GCLinearInput` into physical units.

Beta