#WWDC19

Supporting New Game Controllers

James Kelly, Software Engineer

© 2019 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

Game Controller Framework

Game Controller Framework

Enable support for game controllers

Game Controller Framework

Enable support for game controllers

- iOS, tvOS, and macOS

Game Controller Framework

Enable support for game controllers

- iOS, tvOS, and macOS
- Abstracts hardware through common API

Game Controller Framework

Enable support for game controllers

- iOS, tvOS, and macOS
- Abstracts hardware through common API
- Lets you write your controller code once

NEW

NEW

New Controller Support

NEW

New Controller Support

NEW

Great Games with Controllers

Great Games with Controllers

Rayman Adventures

Great Games with Controllers

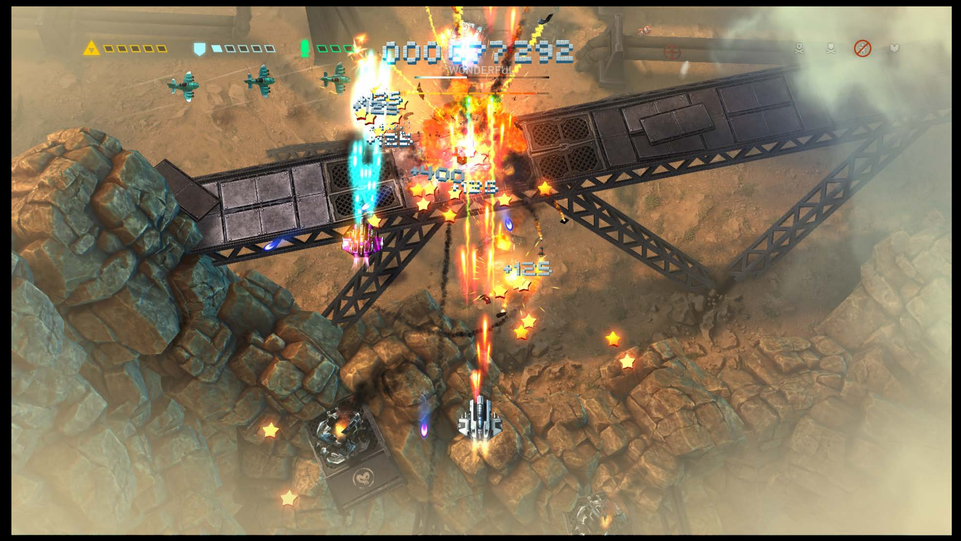
Rayman Adventures

Transistor

Great Games with Controllers

Rayman Adventures

Transistor

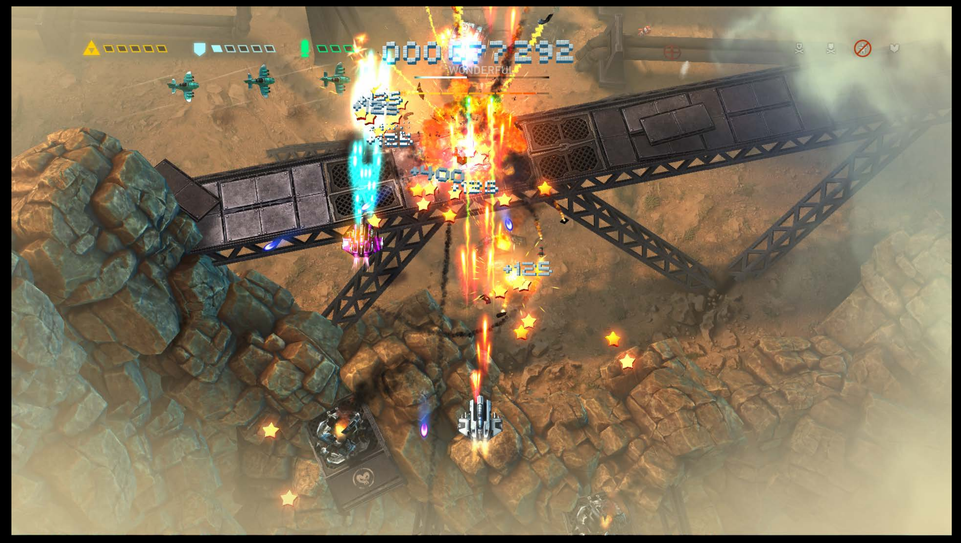


Sky Force Reloaded

Great Games with Controllers

Rayman Adventures

Transistor



Sky Force Reloaded

Alto's Adventure

New Controllers

Accessing Controller Input

UI Best Practices

Legacy macOS Support

New Controllers

New Controllers

New Controllers

Games that already use the Game Controller framework gain support for free

- Xbox Wireless Controller
- DualShock 4

New Controllers

Games that already use the Game Controller framework gain support for free

- Xbox Wireless Controller
- DualShock 4

Inputs accessible via `GCController's GCExtendedGamepad` profile

New Controllers

How to detect controllers

New Controllers

How to detect controllers

GCController represents any physical controller

New Controllers

How to detect controllers

GCController represents any physical controller

Query for connected controllers on app launch

```
GCController.controllers() // Array of currently connected controllers
```

New Controllers

How to detect controllers

GCController represents any physical controller

Query for connected controllers on app launch

```
GCController.controllers() // Array of currently connected controllers
```

Listen for connect and disconnect notifications

```
NSNotification.Name.GCControllerDidConnect
NSNotification.Name.GCControllerDidDisconnect
```

Accessing Controller Input

Accessing Controller Input

Accessing Controller Input

Two Triggers

Accessing Controller Input

Two Triggers

Directional Pad

Accessing Controller Input

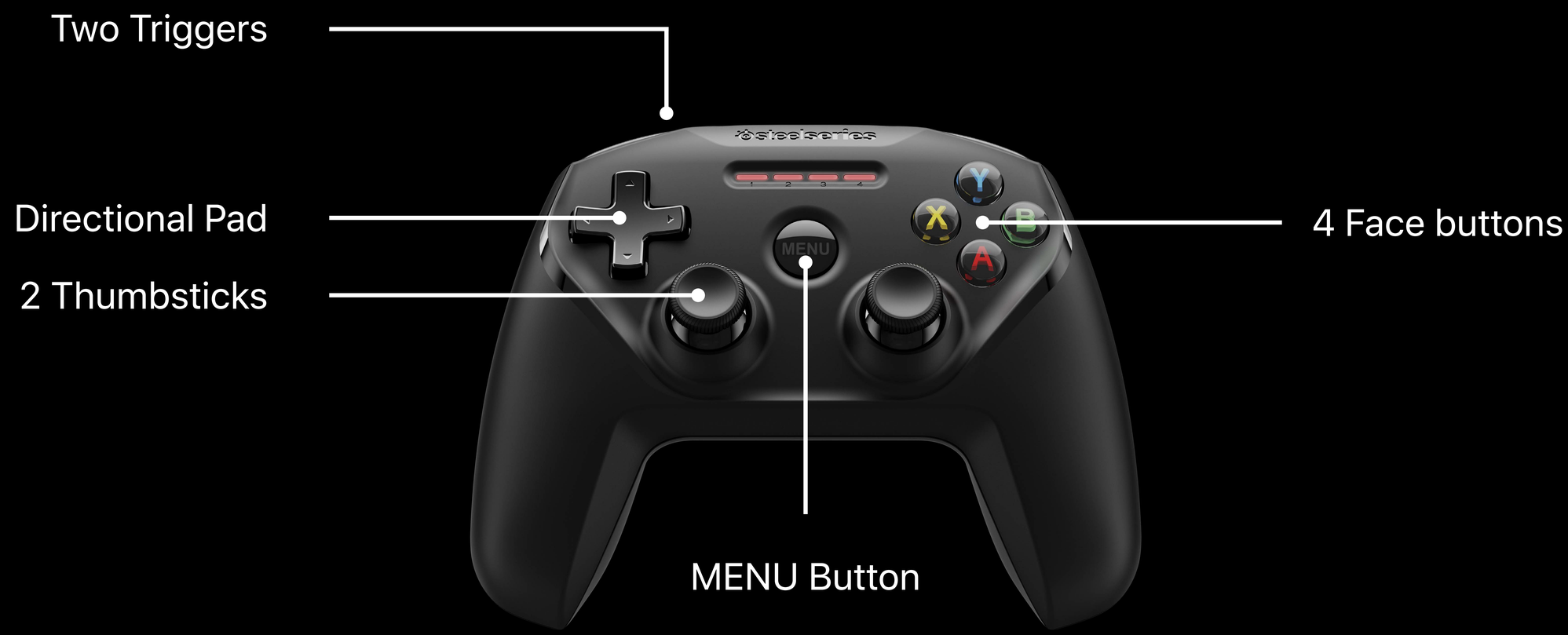
Two Triggers

Directional Pad

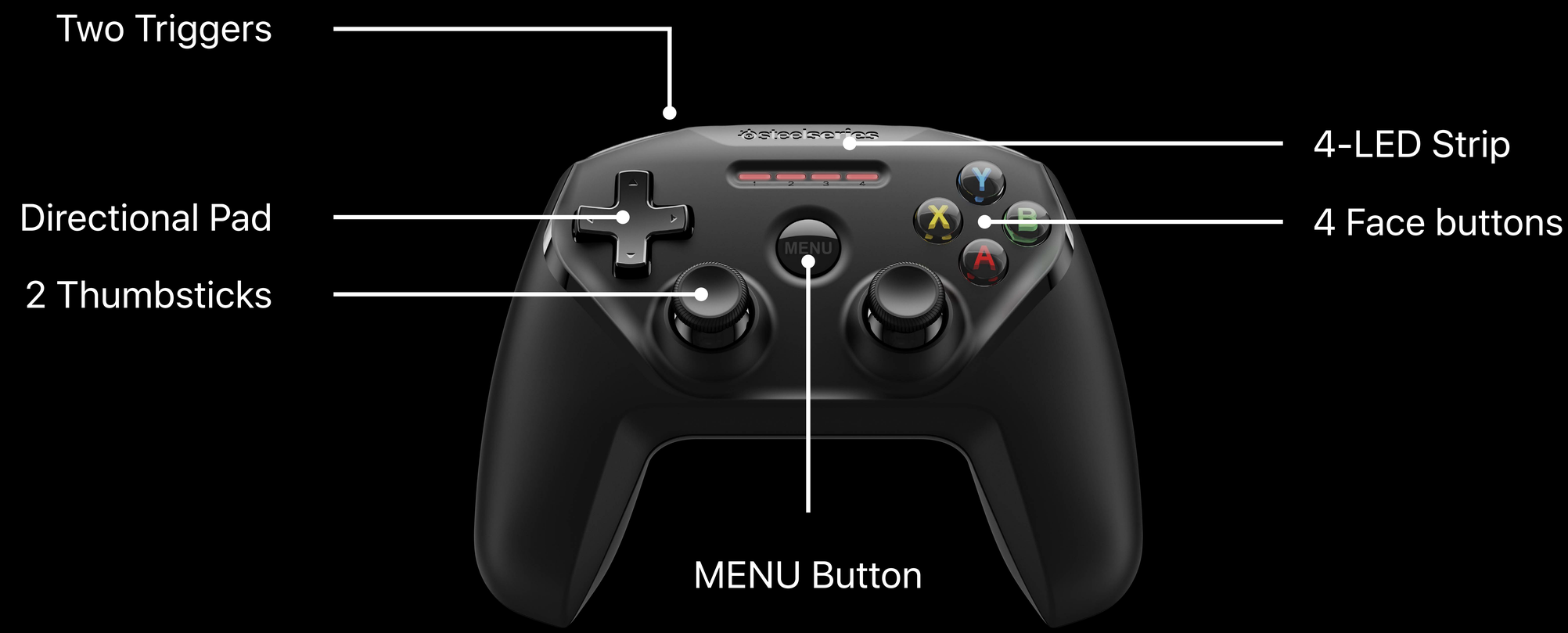
2 Thumbsticks

Accessing Controller Input

Accessing Controller Input



Accessing Controller Input



Accessing Controller Input

Accessing Controller Input

Face buttons

	MFi Controller
`controller.extendedGamepad.buttonA`	A Button
`controller.extendedGamepad.buttonB`	B Button
`controller.extendedGamepad.buttonX`	X Button
`controller.extendedGamepad.buttonY`	Y Button

Accessing Controller Input

Face buttons

	MFi Controller
`controller.extendedGamepad.buttonA`	A Button
`controller.extendedGamepad.buttonB`	B Button
`controller.extendedGamepad.buttonX`	X Button
`controller.extendedGamepad.buttonY`	Y Button

Accessing Controller Input

Face buttons

	Xbox Wireless Controller
`controller.extendedGamepad.buttonA`	A Button
`controller.extendedGamepad.buttonB`	B Button
`controller.extendedGamepad.buttonX`	X Button
`controller.extendedGamepad.buttonY`	Y Button

Accessing Controller Input

Face buttons

	DualShock 4
`controller.extendedGamepad.buttonA`	Cross Button
`controller.extendedGamepad.buttonB`	Circle Button
`controller.extendedGamepad.buttonX`	Square Button
`controller.extendedGamepad.buttonY`	Triangle Button

Accessing Controller Input

Face buttons

	DualShock 4
`controller.extendedGamepad.buttonA`	Cross Button
`controller.extendedGamepad.buttonB`	Circle Button
`controller.extendedGamepad.buttonX`	Square Button
`controller.extendedGamepad.buttonY`	Triangle Button

Accessing Controller Input

Face buttons

	DualShock 4
`controller.extendedGamepad.buttonA`	Cross Button
`controller.extendedGamepad.buttonB`	Circle Button
`controller.extendedGamepad.buttonX`	Square Button
`controller.extendedGamepad.buttonY`	Triangle Button

Accessing Controller Input

Face buttons

	DualShock 4
`controller.extendedGamepad.buttonA`	Cross Button
`controller.extendedGamepad.buttonB`	Circle Button
`controller.extendedGamepad.buttonX`	Square Button
`controller.extendedGamepad.buttonY`	Triangle Button

Accessing Controller Input

Clickable thumbsticks - L3/R3

Accessing Controller Input

Clickable thumbsticks - L3/R3

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Thumbstick Code Example

func registerThumbsticks(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 12.1, tvOS 12.1, macOS 10.14.1, *) {
        guard let leftThumbstickButton = extendedGamepad.leftThumbstickButton else {
            // ensure an alternative path is available
            return
        }
        leftThumbstickButton.pressedChangedHandler = { (button, value, pressed) -> () in
                if pressed { self.togglePlayerCrouch() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

Accessing Controller Input

Auxiliary buttons

Accessing Controller Input

Auxiliary buttons

NEW

Accessing Controller Input

Auxiliary buttons - Menu button

NEW

Accessing Controller Input

Auxiliary buttons - Options button

NEW

Accessing Controller Input

Auxiliary buttons

NEW

	MFi Controller
`controller.extendedGamepad.buttonMenu`	MENU button
`controller.extendedGamepad.buttonOptions`	-

Accessing Controller Input

Auxiliary buttons

NEW

	MFi Controller
`controller.extendedGamepad.buttonMenu`	MENU button
`controller.extendedGamepad.buttonOptions`	-

Accessing Controller Input

Auxiliary buttons

NEW

	MFi Controller
`controller.extendedGamepad.buttonMenu`	MENU button
`controller.extendedGamepad.buttonOptions`	-

Accessing Controller Input

Auxiliary buttons

NEW

	Xbox Wireless Controller
`controller.extendedGamepad.buttonMenu`	Menu button
`controller.extendedGamepad.buttonOptions`	View button

Accessing Controller Input

Auxiliary buttons

NEW

	Xbox Wireless Controller
`controller.extendedGamepad.buttonMenu`	Menu button
`controller.extendedGamepad.buttonOptions`	View button

Accessing Controller Input

Auxiliary buttons

NEW

	Xbox Wireless Controller
`controller.extendedGamepad.buttonMenu`	Menu button
`controller.extendedGamepad.buttonOptions`	View button

Accessing Controller Input

Auxiliary buttons

NEW

	Xbox Wireless Controller
`controller.extendedGamepad.buttonMenu`	Menu button
`controller.extendedGamepad.buttonOptions`	View button

Accessing Controller Input

Auxiliary buttons

NEW

	Xbox Wireless Controller
`controller.extendedGamepad.buttonMenu`	Menu button
`controller.extendedGamepad.buttonOptions`	View button

Accessing Controller Input

Auxiliary buttons

NEW

DualShock 4

`controller.extendedGamepad.buttonMenu`	Options button
`controller.extendedGamepad.buttonOptions`	Share button

Accessing Controller Input

Auxiliary buttons

NEW

	DualShock 4
`controller.extendedGamepad.buttonMenu`	Options button
`controller.extendedGamepad.buttonOptions`	Share button

Accessing Controller Input

Auxiliary buttons

NEW

	DualShock 4
`controller.extendedGamepad.buttonMenu`	Options button
`controller.extendedGamepad.buttonOptions`	Share button

Accessing Controller Input

Auxiliary buttons

NEW

	DualShock 4
`controller.extendedGamepad.buttonMenu`	Options button
`controller.extendedGamepad.buttonOptions`	Share button

Accessing Controller Input

Auxiliary buttons

NEW

	DualShock 4
`controller.extendedGamepad.buttonMenu`	Options button
`controller.extendedGamepad.buttonOptions`	Share button

Accessing Controller Input

Auxiliary buttons - pause handler

Accessing Controller Input

Auxiliary buttons - pause handler

GCController’s controllerPausedHandler has been deprecated

Accessing Controller Input

Auxiliary buttons - pause handler

GCController’s controllerPausedHandler has been deprecated 

- Use buttonMenu API instead

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

```swift
// Pause Handler Code Example

func registerMenuHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        extendedGamepad.buttonMenu.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.togglePause() }
        }
    } else {
        extendedGamepad.controller?.controllerPausedHandler = { _ -> () in
            self.togglePause()
        }
    }
}
```

Accessing Controller Input

Auxiliary buttons - Menu button

Accessing Controller Input

Auxiliary buttons - Options button

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```swift
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

```
// Game Options Code Example

func registerOptionsHandler(extendedGamepad : GCExtendedGamepad) {
    if #available(iOS 13, tvOS 13, macOS 10.15, *) {
        guard let buttonOptions = extendedGamepad.buttonOptions else {
            // ensure an alternative path is available
            return
        }
        buttonOptions.pressedChangedHandler = { (_, _, pressed) -> () in
            if pressed { self.toggleGameOptions() }
        }
    } else {
        // ensure an alternative path is available
    }
}
```

UI Best Practices

UI Best Practices

Customizing in-game guidance

Press B to block

UI Best Practices

Customizing in-game guidance

Press B to block

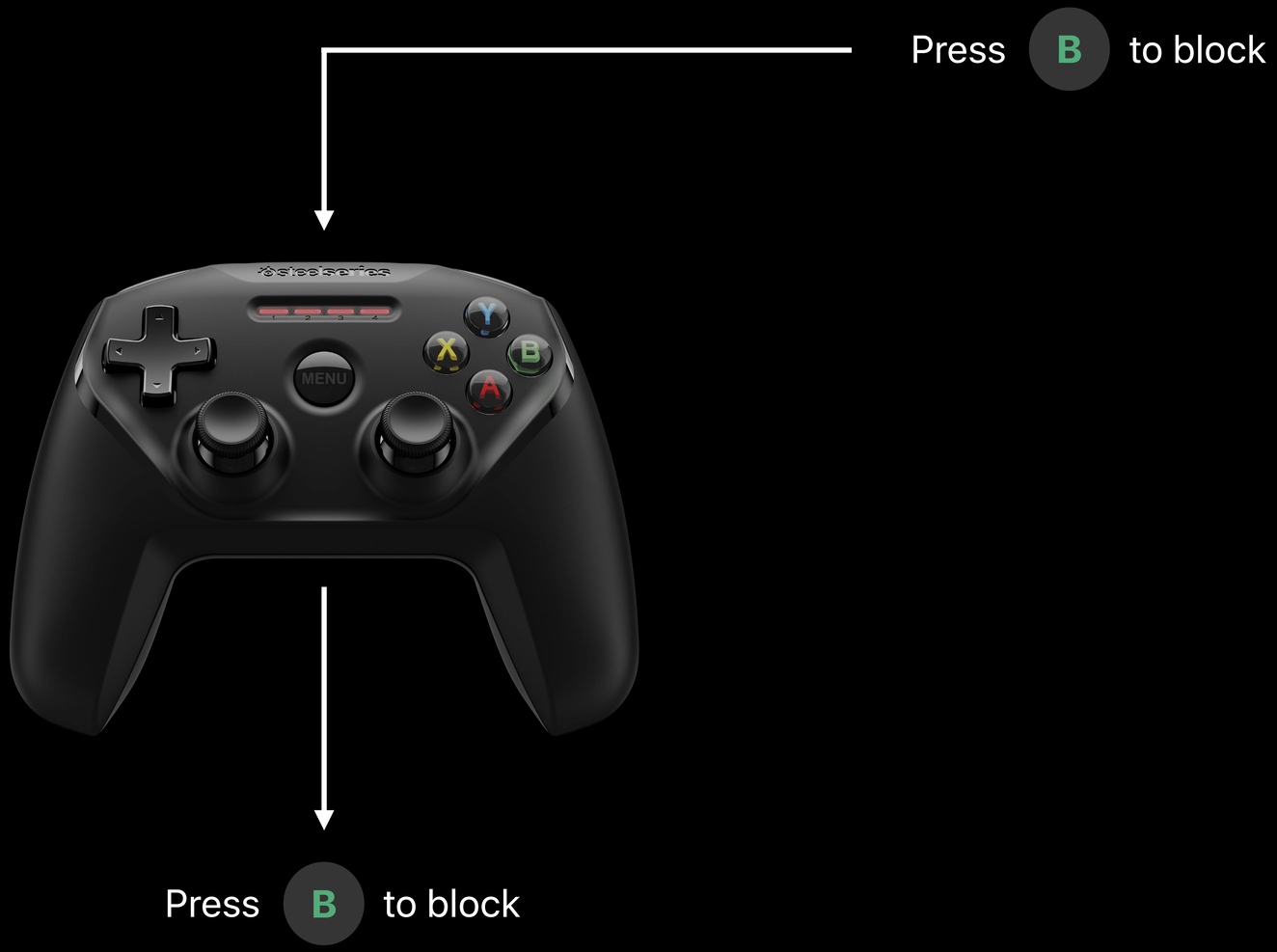
UI Best Practices

Adapt visuals to controller

Press B to block

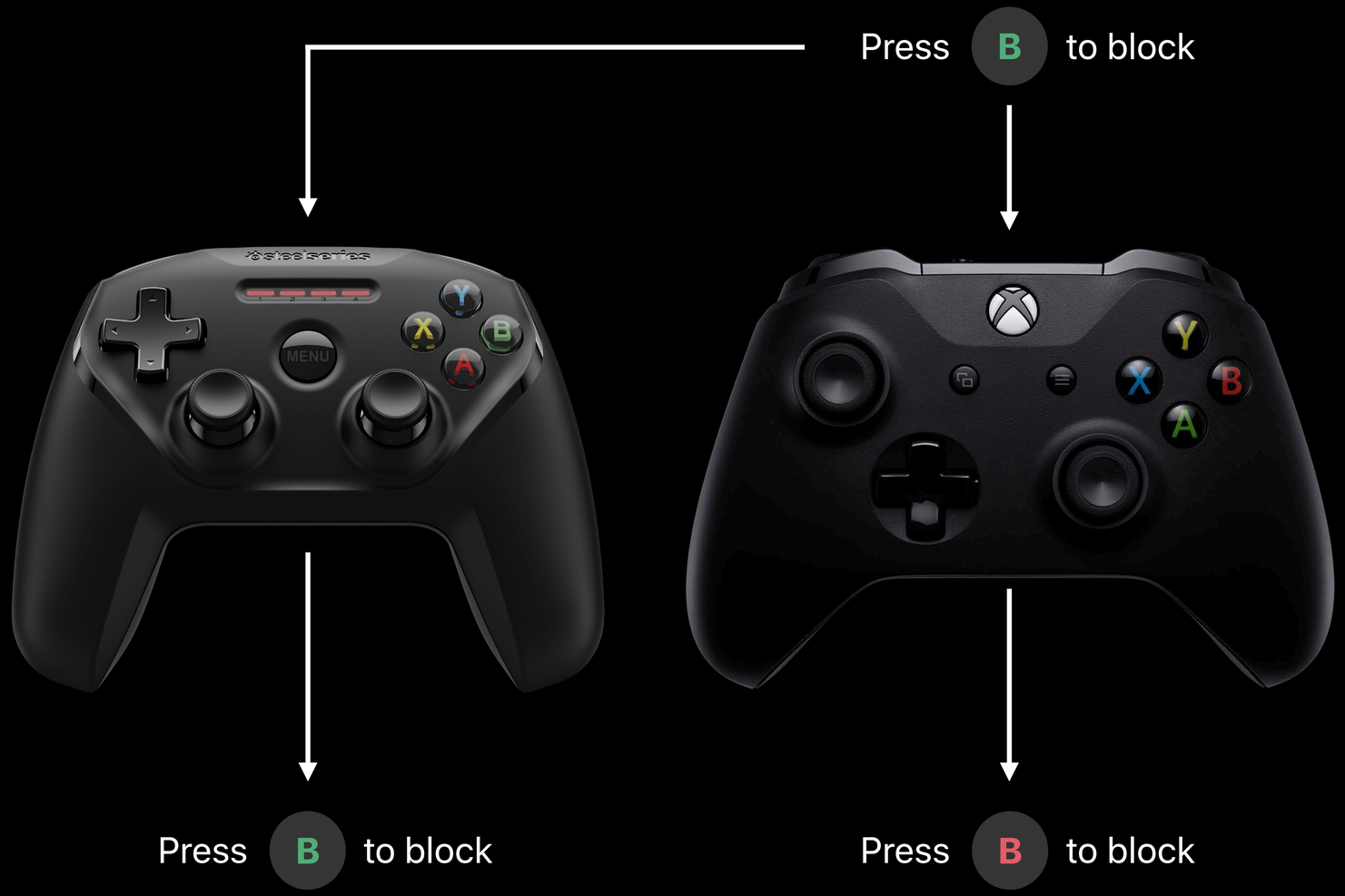
UI Best Practices

Adapt visuals to controller



UI Best Practices

Adapt visuals to controller



UI Best Practices

Adapt visuals to controller

Press B to block

Press B to block

Press B to block

Press ○ to block

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

```
// Controller Product Category Code Example

func getBlockButtonAsset(forController controller: GCController) -> UIImage {
    switch controller.productCategory {
        case "Xbox One":
            return getXboxBButtonAsset(action)
        case "DualShock 4":
            return getDualShock4CircleButtonAsset(action)
        default:
            return getMFiBButtonAsset(action)
    }
}
```

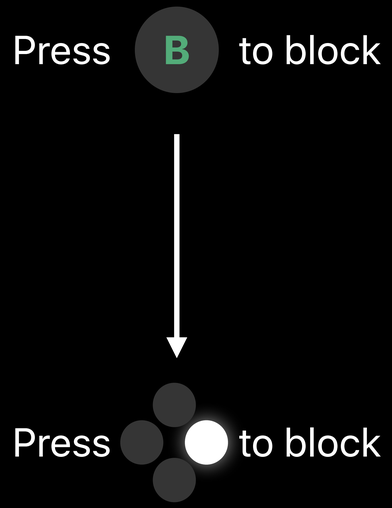
UI Best Practices

Use generic symbols

Press B to block

UI Best Practices

Use generic symbols



UI Best Practices

Adapt to active controller

UI Best Practices

Adapt to active controller

Multiple controllers can be connected at the same time

UI Best Practices

Adapt to active controller

Multiple controllers can be connected at the same time

If your game is single player, gracefully support all connected controllers

UI Best Practices

Adapt to active controller

Multiple controllers can be connected at the same time

If your game is single player, gracefully support all connected controllers

- Recognize input from all connected controllers

UI Best Practices

Adapt to active controller

Multiple controllers can be connected at the same time

If your game is single player, gracefully support all connected controllers

- Recognize input from all connected controllers
- Adapt in-game guidance and visuals to most recently used controller

UI Best Practices

Adapt to active controller

Multiple controllers can be connected at the same time

If your game is single player, gracefully support all connected controllers

- Recognize input from all connected controllers
- Adapt in-game guidance and visuals to most recently used controller

UI Best Practices

Adapt to active controller

Multiple controllers can be connected at the same time

If your game is single player, gracefully support all connected controllers

- Recognize input from all connected controllers
- Adapt in-game guidance and visuals to most recently used controller

Press B to block

DualShock 4 thumbstick moved

Press ○ to block

Legacy macOS Support

Legacy macOS Support

Legacy macOS Support

Use Game Controller framework for supported controllers

Legacy macOS Support

Use Game Controller framework for supported controllers

- Abstracts hardware differences

Legacy macOS Support

Use Game Controller framework for supported controllers

- Abstracts hardware differences
- Provides consistency across platforms

Legacy macOS Support

Use Game Controller framework for supported controllers

- Abstracts hardware differences
- Provides consistency across platforms
- Future proof

Legacy macOS Support

Use Game Controller framework for supported controllers

- Abstracts hardware differences
- Provides consistency across platforms
- Future proof

Migrate hardcoded IOKit implementations

Legacy macOS Support

Use Game Controller framework for supported controllers

- Abstracts hardware differences
- Provides consistency across platforms
- Future proof

Migrate hardcoded IOKit implementations

- New controllers will appear in both IOKit and the Game Controller framework

Legacy macOS Support

Use Game Controller framework for supported controllers

- Abstracts hardware differences
- Provides consistency across platforms
- Future proof

Migrate hardcoded IOKit implementations

- New controllers will appear in both IOKit and the Game Controller framework
- Code to a single interface

Summary

Summary

Summary

Game Controller framework abstracts hardware through a common API

Summary

Game Controller framework abstracts hardware through a common API

Automatically supports newly added controllers

- Xbox Wireless Controller
- DualShock 4

Summary

Game Controller framework abstracts hardware through a common API

Automatically supports newly added controllers

- Xbox Wireless Controller
- DualShock 4

Adapt user interface to match the active controller

Summary

Game Controller framework abstracts hardware through a common API

Automatically supports newly added controllers

- Xbox Wireless Controller
- DualShock 4

Adapt user interface to match the active controller

Future-proof apps on macOS by adopting the Game Controller framework

More Information

https://developer.apple.com/documentation/gamecontroller


WWDC19