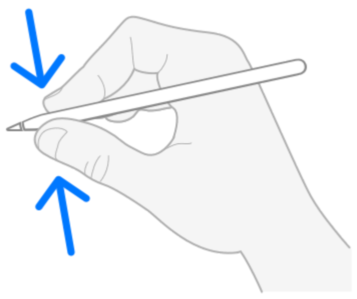
Article

Handling squeezes from Apple Pencil

Detect and respond to squeezes a person makes on Apple Pencil Pro.

Overview

You can use Apple Pencil interactions to allow people to access functionality in your app quickly. Squeezing Apple Pencil Pro lets a person perform actions such as showing a contextual palette without moving the pencil to another location on the screen.



> Note
>
> Only Apple Pencil Pro supports squeeze interactions.

Register for a squeeze

To respond to squeezes from Apple Pencil Pro in your app, you need to register your view to receive squeeze interactions.

SwiftUI UIKit

Add an onPencilSqueeze(perform:) view modifier to your view.

```
MyView()
    .onPencilSqueeze { phase in
        // ...
    }
```

Check the preferred squeeze action

A person can choose which action they prefer to perform when they squeeze Apple Pencil Pro. They choose this preference in Settings > Apple Pencil > Actions > Squeeze. In addition to drawing-specific actions like switching drawing tools, people can configure the preferred action for squeeze to perform any App Shortcut, including pre-configured shortcuts you provide for your app.

In your app, you can check the value of this preferred action for squeeze.

SwiftUI UIKit

To check the preferred action, use the preferredPencilSqueezeAction environment value. For possible values, see PencilPreferredAction.

```
@Environment(\.preferredPencilSqueezeAction) private var preferredAction
```

Choose the action to perform

When possible, perform the preferred action to provide a consistent user experience across apps that support squeezes. If the preferred action doesn't make sense in your app, consider giving people a way to specify a custom action that's suitable for your app. For design guidance, read Human Interface Guidelines > Apple Pencil and Scribble > Squeeze.

The following code shows a snippet from a drawing app that provides custom drawing tools. This app allows a person to configure a custom action to quickly swap to their favorite custom drawing tool instead of using the systemwide preferred action for squeezes. This app also supports the preferred actions to ignore squeezes, switch to the previous tool, and show a custom contextual palette near the Apple Pencil Pro tip.

SwiftUI UIKit

```swift
enum Tool {
    case brush
    case lasso
    case eraser
    case magnifier
}

enum CustomAction: String {
    case switchLasso
    case switchMagnifier
}

@State private var currentTool: Tool? = .brush
@State private var previousTool: Tool?
@State private var contextualPalettePresented = false
@State private var contextualPaletteAnchor: PopoverAttachmentAnchor?

@Environment(\.preferredPencilSqueezeAction) private var preferredAction
@AppStorage("customPencilSqueezeAction") private var customAction: CustomAction?

var body: some View {
    MyView()
        .onPencilSqueeze { phase in
            // Respect the systemwide preferred action to ignore squeezes.
            guard preferredAction != .ignore else { return }

            // If the person chooses to override the systemwide
            // squeeze action to perform a custom action in this app,
            // check which custom action they prefer and perform that action.
            if let customAction, case let .ended(value) = phase {
                if customAction == .switchLasso, currentTool != .lasso {
                    (currentTool, previousTool) = (.lasso, currentTool)
                }
                else if customAction == .switchMagnifier, currentTool != .magnifier
                    (currentTool, previousTool) = (.magnifier, currentTool)
                }
            }

            // If the person prefers to use the systemwide squeeze action,
            // perform the actions that are appropriate in the context of this app:
            // show a custom palette near the Apple Pencil Pro tip, or switch to the
            if preferredAction == .showContextualPalette, case let .ended(value) = p
                if let anchorPoint = value.hoverPose?.anchor {
                    contextualPaletteAnchor = .point(anchorPoint)
                }
                contextualPalettePresented = true
            }
            else if preferredAction == .switchPrevious, case let .ended(value) = pha
                (currentTool, previousTool) = (previousTool, currentTool)
            }
        }
        .popover(
            isPresented: $contextualPalettePresented,
            attachmentAnchor: contextualPaletteAnchor ?? .point(.center)
        ) {
            MyContextualPalette()
        }
}
```

See Also

Related articles

Handling double taps from Apple Pencil

Detect and respond to double taps a person makes on Apple Pencil.

Related reference in SwiftUI

`nonisolated func onPencilSqueeze(perform action: @escaping (PencilSqueezeGesturePhase) -> Void) -> some View`

Adds an action to perform when the user squeezes their Apple Pencil.

`@frozen enum PencilSqueezeGesturePhase`

Describes the phase and value of an Apple Pencil squeeze gesture.

`struct PencilSqueezeGestureValue`

Describes the value of an Apple Pencil squeeze gesture.

`struct PencilPreferredAction`

An action that the user prefers to perform after double-tapping their Apple Pencil.

`struct PencilHoverPose`

A value describing the location and distance of an Apple Pencil hovering in the area above a view’s bounds.

Related reference in UIKit

`@MainActor class UIPencilInteraction`

An interaction that tells your app when a person double-taps or squeezes Apple Pencil.

`@MainActor protocol UIPencilInteractionDelegate : NSObjectProtocol`

The interface an object implements to handle double taps or squeezes a person makes on Apple Pencil.

`@MainActor class Squeeze`

An interaction that represents a squeeze on Apple Pencil.

`enum Phase`

Constants that describe the phases of an interaction on Apple Pencil.

`@MainActor class UIPencilHoverPose`

An object that describes the hover pose of Apple Pencil during an interaction like double tap or squeeze.