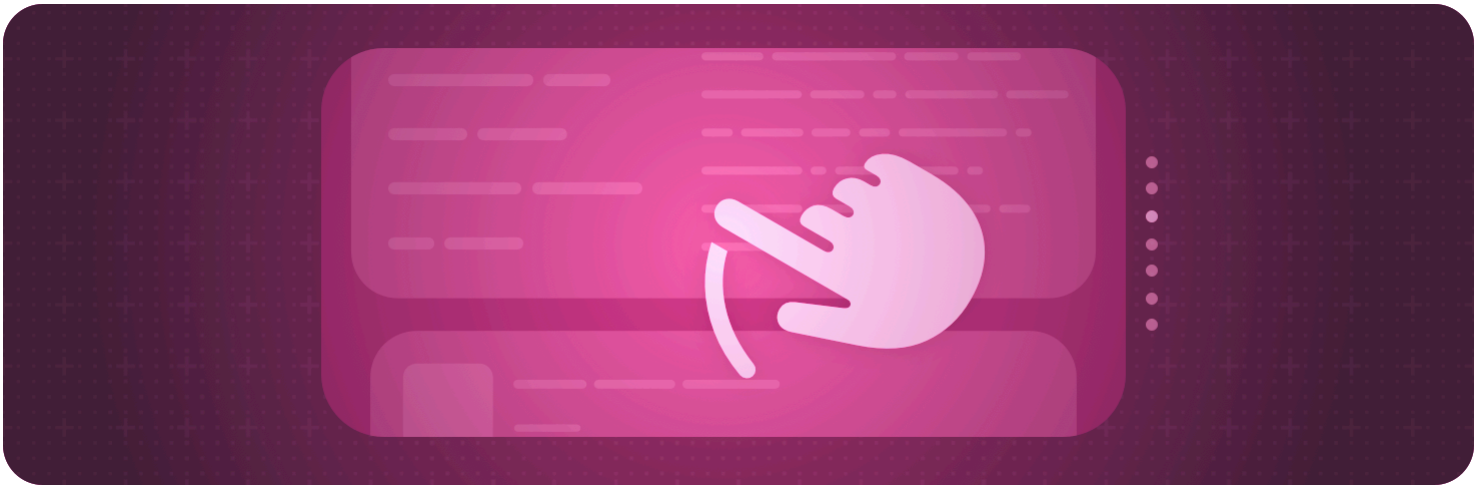
API Collection

Gestures

Define interactions from taps, clicks, and swipes to fine-grained gestures.

Overview

Respond to gestures by adding gesture modifiers to your views. You can listen for taps, drags, pinches, and other standard gestures.



You can also compose custom gestures from individual gestures using the `simultaneously(with:)`, `sequenced(before:)`, or `exclusively(before:)` modifiers, or combine gestures with keyboard modifiers using the `modifiers(_:)` modifier.

Important

When you need a button, use a `Button` instance rather than a tap gesture. You can use any view as the button's label, and the button type automatically provides many of the standard behaviors that users expect from a button, like accessibility labels and hints.

For design guidance, see Gestures in the Human Interface Guidelines.

Topics

Essentials

Adding interactivity with gestures

Use gesture modifiers to add interactivity to your app.

Recognizing tap gestures

```
func onTapGesture(count: Int, perform: () -> Void) -> some View
```

Adds an action to perform when this view recognizes a tap gesture.

```
func onTapGesture(count:coordinateSpace:perform:)
```

Adds an action to perform when this view recognizes a tap gesture, and provides the action with the location of the interaction.

```
struct TapGesture
```

A gesture that recognizes one or more taps.

```
struct SpatialTapGesture
```

A gesture that recognizes one or more taps and reports their location.

Recognizing long press gestures

```
func onLongPressGesture(minimumDuration: Double, maximumDistance: CGFloat, perform: () -> Void, onPressingChanged: ((Bool) -> Void)?) -> some View
```

Adds an action to perform when this view recognizes a long press gesture.

```
func onLongPressGesture(minimumDuration: Double, perform: () -> Void, onPressingChanged: ((Bool) -> Void)?) -> some View
```

Adds an action to perform when this view recognizes a long press gesture.

```
func onLongTouchGesture(minimumDuration: Double, perform: () -> Void, onTouchingChanged: ((Bool) -> Void)?) -> some View
```

Adds an action to perform when this view recognizes a remote long touch gesture. A long touch gesture is when the finger is on the remote touch surface without actually pressing.

```
struct LongPressGesture
```

A gesture that succeeds when the user performs a long press.

Recognizing spatial events

`struct SpatialEventGesture`

A gesture that provides information about ongoing spatial events like clicks and touches.

`struct SpatialEventCollection`

A collection of spatial input events that target a specific view.

`enum Chirality`

The chirality, or handedness, of a pose.

Recognizing gestures that change over time

`func gesture(_:)`

Attaches an NSGestureRecognizerRepresentable to the view.

`func gesture<T>(T, isEnabled: Bool) -> some View`

Attaches a gesture to the view with a lower precedence than gestures defined by the view.

`func gesture<T>(T, name: String, isEnabled: Bool) -> some View`

Attaches a gesture to the view with a lower precedence than gestures defined by the view.

`func gesture<T>(T, including: GestureMask) -> some View`

Attaches a gesture to the view with a lower precedence than gestures defined by the view.

`struct DragGesture`

A dragging motion that invokes an action as the drag-event sequence changes.

`struct WindowDragGesture`

A gesture that recognizes the motion of and handles dragging a window.

`struct MagnifyGesture`

A gesture that recognizes a magnification motion and tracks the amount of magnification.

`struct RotateGesture`

A gesture that recognizes a rotation motion and tracks the angle of the rotation.

`struct RotateGesture3D`

A gesture that recognizes 3D rotation motion and tracks the angle and axis of the rotation.

`struct GestureMask`

Options that control how adding a gesture to a view affects other gestures recognized by the view and its subviews.

Recognizing Apple Pencil gestures

`func onPencilDoubleTap(perform: (PencilDoubleTapGestureValue) -> Void) -> some View`

Adds an action to perform after the user double-taps their Apple Pencil.

`func onPencilSqueeze(perform: (PencilSqueezeGesturePhase) -> Void) -> some View`

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

`var preferredPencilDoubleTapAction: PencilPreferredAction`

The action that the user prefers to perform after double-tapping their Apple Pencil, as selected in the Settings app.

`var preferredPencilSqueezeAction: PencilPreferredAction`

The action that the user prefers to perform when squeezing their Apple Pencil, as selected in the Settings app.

`struct PencilPreferredAction`

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

`struct PencilDoubleTapGestureValue`

Describes the value of an Apple Pencil double-tap gesture.

`struct PencilSqueezeGestureValue`

Describes the value of an Apple Pencil squeeze gesture.

`enum PencilSqueezeGesturePhase`

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

`struct PencilHoverPose`

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

Combining gestures

Composing SwiftUI gestures

Combine gestures to create complex interactions.

`func simultaneousGesture<T>(T, including: GestureMask) -> some View`

Attaches a gesture to the view to process simultaneously with gestures defined by the view.

`func simultaneousGesture<T>(T, isEnabled: Bool) -> some View`

Attaches a gesture to the view to process simultaneously with gestures defined by the view.

`func simultaneousGesture<T>(T, name: String, isEnabled: Bool) -> some View`

Attaches a gesture to the view to process simultaneously with gestures defined by the view.

`struct SequenceGesture`

A gesture that's a sequence of two gestures.

`struct SimultaneousGesture`

A gesture containing two gestures that can happen at the same time with neither of them preceding the other.

`struct ExclusiveGesture`

A gesture that consists of two gestures where only one of them can succeed.

Defining custom gestures

`func highPriorityGesture<T>(T, including: GestureMask) -> some View`

Attaches a gesture to the view with a higher precedence than gestures defined by the view.

`func highPriorityGesture<T>(T, isEnabled: Bool) -> some View`

Attaches a gesture to the view with a higher precedence than gestures defined by the view.

`func highPriorityGesture<T>(T, name: String, isEnabled: Bool) -> some View`

Attaches a gesture to the view with a higher precedence than gestures defined by the view.

`func handGestureShortcut(HandGestureShortcut, isEnabled: Bool) -> some View`

Assigns a hand gesture shortcut to the modified control.

`func defersSystemGestures(on: Edge.Set) -> some View`

Sets the screen edge from which you want your gesture to take precedence over the system gesture.

`protocol Gesture`

An instance that matches a sequence of events to a gesture, and returns a stream of values for each of its states.

`struct AnyGesture`

A type-erased gesture.

`struct HandActivationBehavior`

An activation behavior specific to hand-driven input.

`struct HandGestureShortcut`

Hand gesture shortcuts describe finger and wrist movements that the user can perform in order to activate a button or toggle.

Managing gesture state

`struct GestureState`

A property wrapper type that updates a property while the user performs a gesture and resets the property back to its initial state when the gesture ends.

`struct GestureStateGesture`

A gesture that updates the state provided by a gesture's updating callback.

Handling activation events

`func allowsWindowActivationEvents(Bool?) -> some View`

Configures whether gestures in this view hierarchy can handle events that activate the containing window.

Deprecated gestures

~~`struct MagnificationGesture`~~

A gesture that recognizes a magnification motion and tracks the amount of magnification.

Deprecated

~~`struct RotationGesture`~~

A gesture that recognizes a rotation motion and tracks the angle of the rotation.

Deprecated

See Also

Event handling

- Input events

  Respond to input from a hardware device, like a keyboard or a Touch Bar.

- Clipboard

  Enable people to move or duplicate items by issuing Copy and Paste commands.

- Drag and drop

  Enable people to move or duplicate items by dragging them from one location to another.

- Focus

  Identify and control which visible object responds to user interaction.

- System events

  React to system events, like opening a URL.