Sample Code

Adopting hover support for Apple Pencil

Enhance user feedback for your iPadOS app with a hover preview for Apple Pencil input.

Download

iOS 16.1+ | iPadOS 16.1+ | Xcode 14.1+

Overview

This sample shows how to add support for hover gestures with Apple Pencil. The app is a simple drawing tool that allows a person to draw strokes on a blank canvas. When a person hovers the Apple Pencil slightly above the drawing canvas, the app uses hover gestures to render a visual preview of where the Apple Pencil touches down to draw a stroke on the canvas.

Configure the sample code project

Although the sample app's basic functionality is available in Simulator, running the project with physical devices is recommended to demonstrate the full capabilities of the hardware. Run this project with the following devices:

- iPad Pro 11-inch (4th generation) or iPad Pro 12.9-inch (6th generation) running iPadOS 16.1 or later
- Apple Pencil (2nd generation)

To run the sample code project:

- Connect Apple Pencil with the iPad Pro, as described in Connect Apple Pencil with your iPad.
- Connect the iPad Pro to your Mac with a hardware cable.
- Open the sample code project in Xcode.

- In the Scheme menu, choose the connected iPad Pro.
- Run the app, and use Apple Pencil to interact with the app.

Create a gesture recognizer for drawing

The sample project uses a *long-press gesture recognizer*, which reacts when a person presses and holds a touch for a minimum period of time, to draw strokes with Apple Pencil.

The app implements the `DrawGestureRecognizer` subclass, which extends the capabilities of its superclass `UILongPressGestureRecognizer` to track the `currentTouch` and `currentEvent`. These additional properties provide the information necessary to implement high-fidelity drawing.

```
class DrawGestureRecognizer: UILongPressGestureRecognizer {

    weak var currentTouch: UITouch?
    weak var currentEvent: UIEvent?

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent) {
        super.touchesBegan(touches, with: event)
        currentTouch = touches.first
        currentEvent = event
    }

    override func reset() {
        super.reset()
        currentTouch = nil
        currentEvent = nil
    }
}
```

The drawing implementation is in the `drawGesture(_:)` method. During the `UIGestureRecognizer.State.changed` state of the draw gesture, this method attempts to retrieve additional touches associated with the main touch `currentTouch` through `coalescedTouches(for:)`. This extra touch data creates a smoother drawing experience with lower latency and higher precision.

```
case .changed:
    if let drawGR = drawGestureRecognizer,
       let currentTouch = drawGR.currentTouch,
       let currentEvent = drawGR.currentEvent,
```

```
        let touches = currentEvent.coalescedTouches(for: currentTouch) {
         for touch in touches {
             let point = touch.preciseLocation(in: self)
             updatePath(point: point)
         }
     } else {
         updatePath(point: point)
     }
```

To render the strokes on the canvas, the sample calls `updatePath(point: CGPoint)`, which updates the Bezier path associated with the current stroke and renders the visual output to a `CAShapeLayer`.

Create a gesture recognizer for hover preview

The sample project uses a hover gesture recognizer to generate a visual preview of the stroke before Apple Pencil touches down on the iPad screen. A *hover gesture recognizer* reacts when a pointer from a pointing device such as Apple Pencil moves over a user-interface element. When a person hovers the Apple Pencil a short distance above the iPad screen, the app generates a preview.

The sample project creates an instance of `UIHoverGestureRecognizer` to handle the hover gesture.

```
let hoverGesture = UIHoverGestureRecognizer(target: self, action: #selector(hoverGes
```

The hover preview implementation is in the `hoverGesture(_:)` method. This method waits for the draw gesture to end before starting hover preview rendering. This delay ensures that drawing and previewing don't occur at the same time to create overlapping visual effects in the UI.

```
guard !isDrawing else { return }
```

Update the hover preview

The sample changes the opacity, or *alpha*, of the hover preview effect according to the distance the Apple Pencil hovers above the iPad screen. When Apple Pencil is farther from the screen, the preview alpha is lower, which makes the visual effect more subtle. When Apple Pencil is closer to the screen, the preview alpha is higher, which makes the visual effect more prominent.

The sample calculates the preview alpha value in the `hoverGesture(_:)` method using the following values:

- `zOffset`—A property of `UIHoverGestureRecognizer` that reports the current, normalized distance between the Apple Pencil and the iPad screen.
- `maxPreviewZOffset`—A constant that represents the maximum distance between the Apple Pencil and the iPad screen. The sample uses the maximum distance `1.0` because `zOffset` is normalized.
- `fadeZOffset`—A constant that represents the threshold distance at which the preview alpha switches between fully opaque and partially opaque, causing the visual preview effect to begin fading out.

```
previewAlpha = 1.0 - max(zOffset - fadeZOffset, 0.0) / (maxPreviewZOffset - fadeZOf
```

This line of code calculates the preview alpha. When Apple Pencil is closer to the screen than `fadeZOffset`, the preview alpha is `1.0`, which makes the visual effect fully opaque. As Apple Pencil moves farther away than `fadeZOffset`, the alpha begins to decrease and eventually reaches `0.0`, which causes the visual preview to fade and disappear.

Configure different behaviors for input types

By default, hover gestures work with pointing devices such as trackpads as well as Apple Pencil. For example, running the sample code project on an iPad with a connected trackpad renders a hover preview when the pointer passes over the drawing canvas. However, this sample provides a different hover experience for Apple Pencil than for a trackpad, so the default behavior might be unsuitable for trackpad input.

To restrict the hover preview to Apple Pencil input only and disable it for trackpad, uncomment the following line of code and run the app again:

```
// hoverGesture.allowedTouchTypes = [ UITouch.TouchType.pencil.rawValue as NSNumber
```

See Also

Standard gestures

Handling UIKit gestures

Use gesture recognizers to simplify touch handling and create a consistent user experience.

Coordinating multiple gesture recognizers

Discover how to use multiple gesture recognizers on the same view.

{} Supporting gesture interaction in your apps

Enrich your app's user experience by supporting standard and custom gesture interaction.

`class` `UIHoverGestureRecognizer`

A continuous gesture recognizer that interprets pointer movement over a view.

`class` `UILongPressGestureRecognizer`

A continuous gesture recognizer that interprets long-press gestures.

`class` `UIPanGestureRecognizer`

A continuous gesture recognizer that interprets panning gestures.

`class` `UIPinchGestureRecognizer`

A continuous gesture recognizer that interprets pinching gestures involving two touches.

`class` `UIRotationGestureRecognizer`

A continuous gesture recognizer that interprets rotation gestures involving two touches.

`class` `UIScreenEdgePanGestureRecognizer`

A continuous gesture recognizer that interprets panning gestures that start near an edge of the screen.

`class` `UISwipeGestureRecognizer`

A discrete gesture recognizer that interprets swiping gestures in one or more directions.

`class` `UITapGestureRecognizer`

A discrete gesture recognizer that interprets single or multiple taps.