API Collection

Pointer interactions

Support pointer interactions in your custom controls and views.

Overview

iPadOS 13.4 introduces dynamic pointer effects and behaviors that enhance the experience of using an external input device, like a trackpad or mouse, with iPad. As people use an input device, iPadOS automatically adapts the pointer to the current context, providing rich visual feedback and just the right level of precision needed to enhance productivity and simplify common tasks.

UIKit automatically handles pointer interactions if you're using `UIButton`, `UIBarButtonItem`, or `UISegmentedControl`. If you use custom views to display your content, you must define pointer effects and styles yourself.

For more information, see Human Interface Guidelines.

Specify custom pointer styles

To add a custom pointer style effect to a view:

1. Create a `UIPointerInteraction` instance.
2. Specify the pointer interaction's delegate (an object that conforms to the `UIPointerInteractionDelegate` protocol).
3. Add the interaction to the view's `interactions` property.
4. Add the `pointerInteraction(_:styleFor:)` delegate method.
5. Return `UIPointerStyle` from that delegate method.

This example uses a custom helper method, which you typically call within a view controller's `viewDidLoad()` method:

```
func customPointerInteraction(on view: UIView, pointerInteractionDelegate: UIPointer
    let pointerInteraction = UIPointerInteraction(delegate: pointerInteractionDelega
    view.addInteraction(pointerInteraction)
}
```

The pointerInteraction(_:styleFor:) delegate method is called when the pointer enters the view's region. The following example shows an interaction that applies a UIPointerLiftEffect effect by returning a UIPointerStyle object:

```
func pointerInteraction(_ interaction: UIPointerInteraction, styleFor region: UIPoir
    var pointerStyle: UIPointerStyle? = nil

    if let interactionView = interaction.view {
        let targetedPreview = UITargetedPreview(view: interactionView)
        pointerStyle = UIPointerStyle(effect: UIPointerEffect.lift(targetedPreview))
    }
    return pointerStyle
}
```

Add interaction animations

Including animations can be helpful in pointer interactions, especially when views contain elements that interfere with pointer effects. For example, hiding the separator bars in a UISegmentedControl when the pointer enters the control allows the active segment effect to appear visually uncluttered.

The following example performs a simple animation to change the alpha value of the view when the pointer enters and exits the region:

```
func pointerInteraction(_ interaction: UIPointerInteraction, willEnter region: UIPoi
    if let interactionView = interaction.view {
        animator.addAnimations {
            interactionView.alpha = 0.5
        }
    }
}

func pointerInteraction(_ interaction: UIPointerInteraction, willExit region: UIPoir
    if let interactionView = interaction.view {
        animator.addAnimations {
            interactionView.alpha = 1.0
        }
    }
}
```

Distinguish pointing device input events

If you want to distinguish between pointing device touch events and touch events from other sources, like the user's fingers or Apple Pencil, you can enable the `UIApplicationSupportsIndirectInputEvents` key in the `Info.plist` file. With this key enabled, your app can respond to specific gestures targeted at touches of `UITouch.TouchType.indirectPointer`.

For more information, see `UIApplicationSupportsIndirectInputEvents`.

Topics

Essentials

`class UIPointerInteraction`

An interaction that enables support for effects on a view or customizes the pointer's appearance within a region of an app.

`protocol UIPointerInteractionDelegate`

An interface for handling pointer movements within the interaction's view.

{} Integrating pointer interactions into your iPad app

Support touch interactions in your iPad app by adding pointer interactions to your views.

{} Enhancing your iPad app with pointer interactions

Provide a great user experience with pointing devices, by incorporating pointer content effects and shape customizations.

Interaction animations

`protocol UIPointerInteractionAnimating`

An interface for modifying an interaction animation in coordination with the pointer effect animations.

Pointer styles

`class UIPointerStyle`

An object that defines the pointer shape and effect.

`enum UIPointerShape`

An object that defines the shape of custom pointers.

`enum UIPointerEffect`

An effect that alters a view’s appearance when a pointer enters the current region.

`class UIPointerAccessory`

Constants that describe accessories to display alongside the primary pointer.

Pointer region

`class UIPointerRegion`

A rectangular region that interacts with pointer movements.

`class UIPointerRegionRequest`

An object to describe the pointer’s location in the interaction’s view.

Lock state

`class UIPointerLockState`

An object that contains information about a scene’s pointer lock state.

Band selection

`class UIBandSelectionInteraction`

An object that tracks the selection of multiple items using pointer-based input.

`enum State`

Constants that indicate whether a band selection interaction object is inactive or currently tracking an interaction.

See Also

User interactions

- Touches, presses, and gestures

  Encapsulate your app’s event-handling logic in gesture recognizers so that you can reuse that code throughout your app.

- Menus and shortcuts

  Simplify interactions with your app using menu systems, contextual menus, Home Screen quick actions, and keyboard shortcuts.

- Drag and drop

  Bring drag and drop to your app by using interaction APIs with your views.

- Apple Pencil interactions

  Handle user interactions like double tap and squeeze on Apple Pencil.

- Focus-based navigation

  Navigate the interface of your UIKit app using a remote, game controller, or keyboard.

- Accessibility for UIKit

  Make your UIKit apps accessible to everyone who uses iOS and tvOS.