PencilKit

Talleres iOS - Semana 18

Alonso Huerta - Dec 3 2024

Antes de comenzar…

Algunos consejos y avisos

- Si tienen un iPad y/o Apple Pencil, úsenlo para este tutorial. (No es necesario)
- Usaremos Playgrounds, pero puede usar un proyecto normal.
- Este es 1 de 6 talleres que vamos a impartir entre hoy (martes 3 de Diciembre) y mañana (miércoles 4 de Diciembre).
- Si necesitan ayuda pueden levantar la mano y uno de nuestros talleristas irá con ustedes.

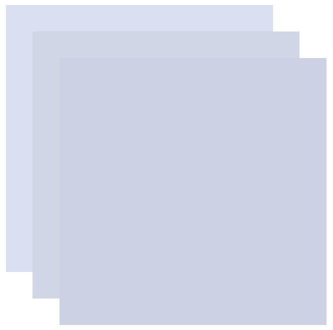
import PencilKit

Framework

PencilKit

Capture touch and Apple Pencil input as a drawing, and display that content from your app.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | visionOS 1.0+



Canvas

{} Drawing with PencilKit

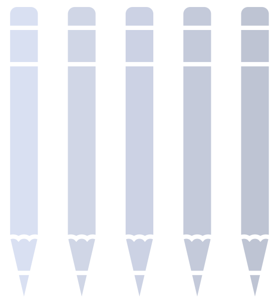
Add expressive, low-latency drawing to your app using PencilKit.

{} Customizing Scribble with Interactions

Enable writing on a non-text-input view by adding interactions.

{} Inspecting, Modifying, and Constructing PencilKit Drawings

Score users' ability to match PencilKit drawings generated from text, by accessing the strokes and points inside PencilKit drawings.



Tools

{} Configuring the PencilKit tool picker

Incorporate a custom PencilKit tool picker with a variety of system and custom tools into a drawing app.

Class

PKCanvasView

A view that captures Apple Pencil input and displays the rendered results in an iOS app.

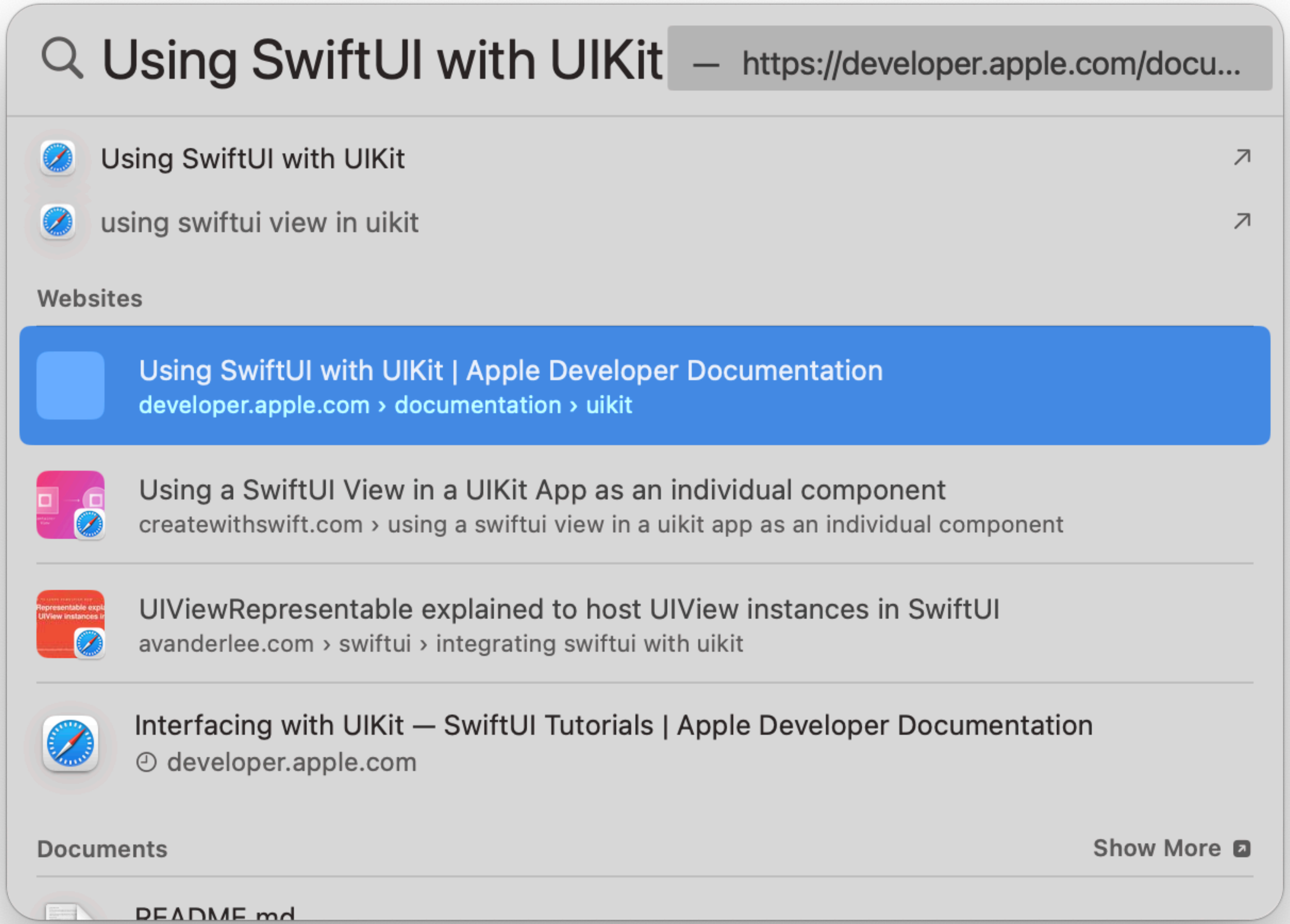
iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.1+ | visionOS 1.0+

```
@MainActor
class PKCanvasView
```

```
struct ContentView: View {
    var body: some View {
        VStack {
            PKCanvasView()
        }
    }
}
```

❌ 'buildExpression' is unavailable: this expression does not conform to 'View'

```
struct ContentView: View {  ← SwiftUI View
    var body: some View {
        VStack {
            PKCanvasView()  ← UIKit View
        }
    }
}
```


Using SwiftUI with UIKit — https://developer.apple.com/docu...

Using SwiftUI with UIKit

using swiftui view in uikit

Websites

Using SwiftUI with UIKit | Apple Developer Documentation
developer.apple.com › documentation › uikit

Using a SwiftUI View in a UIKit App as an individual component
createwithswift.com › using a swiftui view in a uikit app as an individual component

UIViewRepresentable explained to host UIView instances in SwiftUI
avanderlee.com › swiftui › integrating swiftui with uikit

Interfacing with UIKit — SwiftUI Tutorials | Apple Developer Documentation
developer.apple.com

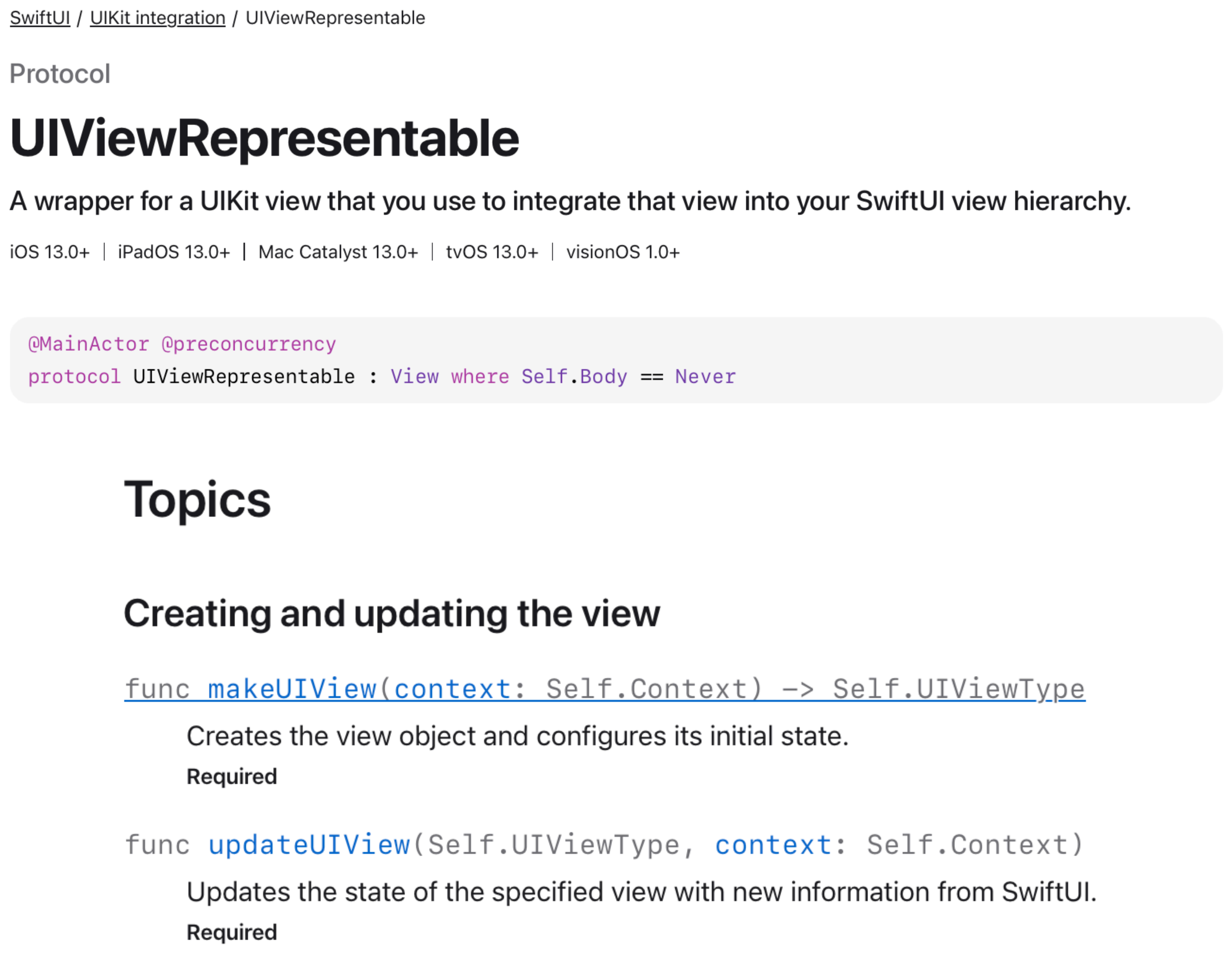
Documents

Show More

README.md

Interfacing with UIKit

Framework integration



SwiftUI / UIKit integration / UIViewRepresentable

Protocol

UIViewRepresentable

A wrapper for a UIKit view that you use to integrate that view into your SwiftUI view hierarchy.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | tvOS 13.0+ | visionOS 1.0+

```
@MainActor @preconcurrency
protocol UIViewRepresentable : View where Self.Body == Never
```

Topics

Creating and updating the view

```
func makeUIView(context: Self.Context) -> Self.UIViewType
```

Creates the view object and configures its initial state.

Required

```
func updateUIView(Self.UIViewType, context: Self.Context)
```

Updates the state of the specified view with new information from SwiftUI.

Required

```swift
struct CanvasView: UIViewRepresentable {

    func makeUIView(context: Context) -> PKCanvasView {
        // Return a Canvas View
        return PKCanvasView()
    }

    func updateUIView(_ uiView: PKCanvasView, context: Context) {
        // TODO
    }

}

struct ContentView: View {
    var body: some View {
        VStack {
            CanvasView()
        }
    }
}
```

Configuring the drawing environment

`var tool: any PKTool`

The currently selected tool used for drawing.

`var isRulerActive: Bool`

A Boolean value that indicates whether a ruler view is visible on the canvas.

~~`var allowsFingerDrawing: Bool`~~

A Boolean value that indicates whether the canvas accepts input from the user's finger in addition to Apple Pencil.

Deprecated

`var drawingPolicy: PKCanvasViewDrawingPolicy`

The policy that controls the types of touches allowed when drawing on the canvas.

`enum PKCanvasViewDrawingPolicy`

Constants that you use to specify the type of drawing gestures your app permits while the user draws on the canvas.

Topics

Drawing policies

⚠️ `case `default``

The default input type to use for drawing on a canvas.

`case anyInput`

Allows drawing on the canvas from any input source.

⚠️ `case pencilOnly`

Pencil touches are the only input that draw on the canvas.

```swift
struct CanvasView: UIViewRepresentable {

    func makeUIView(context: Context) -> PKCanvasView {
        // Return a Canvas View
        let canvas = PKCanvasView()
        canvas.drawingPolicy = .anyInput
        return canvas
    }

    func updateUIView(_ uiView: PKCanvasView, context: Context) {
        // TODO
    }

}
```

PencilKit / PKToolPicker

Class

PKToolPicker

A tool palette that displays a selection of drawing tools and colors for tools that a person can choose from.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.1+ | visionOS 1.0+

```
class PKToolPicker
```

```swift
struct ToolPickerView: UIViewRepresentable {

    func makeUIView(context: Context) -> PKToolPicker {
        // Return a Tool Picker View
        return PKToolPicker()
    }

    func updateUIView(_ uiView: PKToolPicker, context: Context) {
        // TODO
    }

}
```

Candidate would match and infer 'UIViewType' = 'PKToolPicker' if 'PKToolPicker' inherited from 'UIView'

Overview

A PKToolPicker manages a draggable palette that displays drawing tools, colors, and additional options. You add a tool picker to your interface and configure it to display its palette at appropriate times. While the palette is onscreen, a person may reposition it anywhere within the current window. When a person interacts with the palette, the tool picker notifies registered observers of the changes so that they can respond.

Important

The tool picker doesn't display in Mac apps built with Mac Catalyst.

When configuring your interface, call the setVisible(_:forFirstResponder:) method to associate the tool picker with one or more views in your interface. Each window manages its own tool picker, and the window's first responder determines the visibility of that tool picker. When one of the registered objects becomes first responder, the tool picker automatically adds its palette view to the current window. When there isn't a registered object as first responder, the tool picker hides its palette view.

PKCanvasView implements the observer protocol for detecting tool picker changes. Adding your canvas view as an observer to a tool picker automatically updates the current drawing tools. For more information about implementing custom observer objects, see PKToolPickerObserver.

```swift
struct ContentView: View {
    @State var toolPicker = PKToolPicker()

    var body: some View {
        VStack {
            CanvasView()
        }
        .onAppear {
            toolPicker.setVisible(true, forFirstResponder: ???)
        }
    }
}
```

```
func setVisible(
    _ visible: Bool,
    forFirstResponder responder: UIResponder
)
```

Parameters

`visible`
A Boolean value that indicates whether to make the palette visible when `responder` becomes active. Specify `true` to show the palette when the object becomes the first responder.

⚠️ `responder`
A responder object associated with the tool picker’s window. Typically, you specify a view capable of becoming the first responder.

UIKit / Touches, presses, and gestures / UIResponder

Class

UIResponder

An abstract interface for responding to and handling events.

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.1+ | tvOS 9.0+ | visionOS 1.0+

```
@MainActor
class UIResponder : NSObject
```

```swift
struct CanvasView: UIViewRepresentable {
    @Binding var canvas: PKCanvasView

    func makeUIView(context: Context) -> PKCanvasView {
        // Return the Canvas View
        canvas.drawingPolicy = .anyInput
        return canvas
    }

    func updateUIView(_ uiView: PKCanvasView, context: Context) {
        // TODO
    }

}

struct ContentView: View {
    @State var canvas = PKCanvasView()
    @State var toolPicker = PKToolPicker()

    var body: some View {
        VStack {
            CanvasView(canvas: $canvas)
        }
        .onAppear {
            toolPicker.setVisible(true, forFirstResponder: canvas)
            canvas.becomeFirstResponder()
        }
    }
}
```

Detecting changes to the picker

`func addObserver(any PKToolPickerObserver)`

Adds the specified object to the list of objects to notify when the picker configuration changes.

`func removeObserver(any PKToolPickerObserver)`

Removes the specified object from the list of objects to notify when the picker configuration changes.

`protocol PKToolPickerObserver`

An interface you use to detect when the user changes the selected tools and drawing characteristics of a tool picker object.

```swift
struct ContentView: View {
    @State var canvas = PKCanvasView()
    @State var toolPicker = PKToolPicker()

    var body: some View {
        VStack {
            CanvasView(canvas: $canvas)

        }
        .onAppear {
            toolPicker.setVisible(true, forFirstResponder: canvas)
            canvas.becomeFirstResponder()
            toolPicker.addObserver(canvas)
        }
    }
}
```

```
struct CanvasView: UIViewRepresentable {
    @Binding var canvas: PKCanvasView

    func makeUIView(context: Context) -> PKCanvasView {
        // Return the Canvas View
        canvas.drawingPolicy = .anyInput
        return canvas
    }

    func updateUIView(_ uiView: PKCanvasView, context: Context) {
        // NOTHING TO DO HERE :)
    }

}

struct ContentView: View {
    @State var canvas = PKCanvasView()
    @State var toolPicker = PKToolPicker()

    var body: some View {
        VStack {
            CanvasView(canvas: $canvas)
        }
        .onAppear {
            toolPicker.setVisible(true, forFirstResponder: canvas)
            canvas.becomeFirstResponder()
            toolPicker.addObserver(canvas)
        }
    }
}
```

Ahora es tu turno

Ejercicios para PencilKit

- Haz que el Canvas sea más grande que la pantalla.
- Puedes tener más un Canvas en la aplicación.
- Para el ToolKit, solo ciertas herramientas están disponibles.
- Agregar un contador con la cantidad de `strokes` en el Canvas.

Learn Swift

Talleres iOS - Semana 18

Dec 3 2024