Instance Method

translationPresentation(isPresented:text:attachmentAnchor:arrowEdge:replacementAction:)

Presents a translation popover when a given condition is true.

Translation | SwiftUI | iOS 17.4+ | iPadOS 17.4+ | macOS 14.4+

```
nonisolated
func translationPresentation(
    isPresented: Binding<Bool>,
    text: String,
    attachmentAnchor: PopoverAttachmentAnchor = .rect(.bounds),
    arrowEdge: Edge = .top,
    replacementAction: ((String) -> Void)? = nil
) -> some View
```

Parameters

`isPresented`
A binding to a Boolean value that determines whether to present the popover.

`text`
The text to be translated. Changing this after the popover presents has no effect.

`attachmentAnchor`
The positioning anchor that defines the attachment point of the popover. The default is bounds.

`arrowEdge`
The edge of the `attachmentAnchor` that defines the location of the popover's arrow in macOS. The default is `Edge.top`. iOS ignores this parameter.

`replacementAction`
An optional action to perform when someone interacts with the "Replace with Translation" button. The "Replace with Translation" button appears when you provide this action.

Discussion

Use this method to show a system UI translation popover when the `translationVisible` Boolean variable is `true`. Attach this view modifier to the containing view holding the text to translate. Then set the `isPresented` Boolean binding to `true` when you want the popover to display. When you do, a system UI translation popover appears offering a translation.

In the example below, the popover appears when the user toggles the `translationVisible` state variable by pressing the Translate button.

```
struct ContentView: View {
    @State private var translationVisible = false
    private var originalText = "Hallo, Welt!"

    var body: some View {
        VStack {
            Text(verbatim: originalText)
                .translationPresentation(isPresented: $translationVisible, text: ori
            Button("Translate") {
                translationVisible.toggle()
            }
        }
    }
}
```

To replace the original text with the translation, pass in a trailing closure to the `replacement Action` parameter. When you do, a Replace with Translation button appears in the system UI. When the user taps this button, the closure returns a translation of the text as a string. Use that string to update the original text with the translated text as shown below:

```
struct ReplaceTranslation: View {
    @State private var translationVisible = false
```

```
    @State private var originalText = "Hallo, Welt!"

    var body: some View {
        VStack(spacing: 20) {
            TextField("Text to translate", text: $originalText, axis: .vertical)
                .textFieldStyle(.roundedBorder)
                .translationPresentation(isPresented: $translationVisible, text: ori
                    // Update the original text with the translated result.
                    originalText = translatedString
                }
            Button("Translate") {
                translationVisible.toggle()
            }
        }
        .padding()
    }
}
```

If the system can't detect the source language, it asks the user to select the language to translate from. For a list of supported languages see `LanguageAvailability.supportedLanguages`.

While this function does support long strings of text, it works best with short ones (no more than a couple of sentences or phrases in length).

See Also

Showing a translation

`func translationTask(TranslationSession.Configuration?, action: (TranslationSession) async -> Void) -> some View`

Adds a task to perform before this view appears or when the translation configuration changes.

`func translationTask(source: Locale.Language?, target: Locale.Language?, action: (TranslationSession) async -> Void) -> some View`

Adds a task to perform before this view appears or when the specified source or target languages change.