Sample Code

Landmarks: Displaying custom activity badges

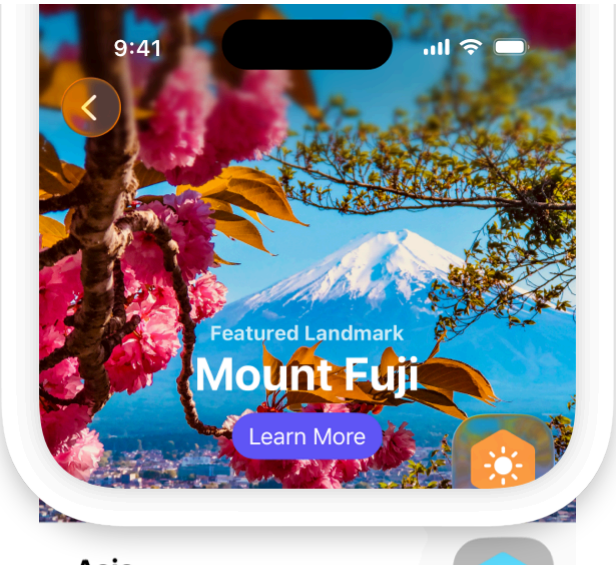
Provide people with a way to mark their adventures by displaying animated custom activity badges.

Download

iOS 26.0+ | iPadOS 26.0+ | macOS 26.0+ | Xcode 26.0+

Overview

The Landmarks app lets people track their adventures as they explore sites around the world. Whether it's a national park near their home or a far-flung location on a different continent, the app provides a way for people to mark their adventures and receive custom activity badges along the way.



This sample displays the badges in a vertical view that includes a toggle button for showing or hiding the badges. The Landmarks app includes a custom modifier that makes it easier for other views to adopt the badge view. By configuring the badges to use Liquid Glass, the badges gain the advantage of using the morphing animation when you show or hide the badges.

Add a modifier to show badges in other views

To make the badges available in other views, like `CollectionsView`, the sample uses a custom modifier, `ShowBadgesViewModifier`, as a `ViewModifier`. The sample layers the badges over another view using a `ZStack`, and positions the badge view in the lower trailing corner:

```
private struct ShowsBadgesViewModifier: ViewModifier {
    func body(content: Content) -> some View {
        ZStack {
            content
            HStack {
                Spacer()
                VStack {
                    Spacer()
                    BadgesView()
                        .padding()
                }
            }
        }
    }
}
```

The sample extends `View` by adding the `showBadges` modifier:

```
extension View {
    func showsBadges() -> some View {
        modifier(ShowsBadgesViewModifier())
    }
}
```

Apply Liquid Glass to the toggle button

To create the toggle button, the sample configures a `Button` using `ToggleBadgesLabel` which has different system images for the Show and Hide toggle states. To apply Liquid Glass, style the button with the `glass` modifier:

```
Button {
    //...
} label: {
    //...
}
.buttonStyle(.glass)
```

Add Liquid Glass to the badges

To add Liquid Glass to each badge, the sample uses the `glassEffect(_:in:)` modifier. To make a custom glass view appearance, the sample specifies a rectangular option with a corner radius:

```
BadgeLabel(badge: $0)
    .glassEffect(.regular, in: .rect(cornerRadius: Constants.badgeCornerRadius))
```

Animate the badges using the morph effect

The morph effect is an animation for Liquid Glass views. During this animation, the toggle button and each badge start as a combined view. Then, the button and badges change shape like a liquid as they separate and move from one location to another. In reverse, the toggle button and badges change shape and combine back into one view.

To achieve the Liquid Glass morph effect, the app:

- organizes the badges and toggle button into a `GlassEffectContainer`

- adds `glassEffectID(_:in:)` to each badge
- adds `glassEffectID(_:in:)` to the toggle button
- wraps the command that toggles the `isExpanded` property in `withAnimation(_:_:)`

```
// Organizes the badges and toggle button to animate together.
GlassEffectContainer(spacing: Constants.badgeGlassSpacing) {
    VStack(alignment: .center, spacing: Constants.badgeButtonTopSpacing) {
        if isExpanded {
            VStack(spacing: Constants.badgeSpacing) {
                ForEach(modelData.earnedBadges) {
                    BadgeLabel(badge: $0)
                        // Adds Liquid Glass to the badge.
                        .glassEffect(.regular, in: .rect(cornerRadius: Constants.bad
                        // Adds an identifier to the badge for animation.
                        .glassEffectID($0.id, in: namespace)
                }
            }
        }

        Button {
            // Animates this button and badges when `isExpanded` changes values.
            withAnimation {
                isExpanded.toggle()
            }
        } label: {
            ToggleBadgesLabel(isExpanded: isExpanded)
                .frame(width: Constants.badgeShowHideButtonWidth,
                       height: Constants.badgeShowHideButtonHeight)
        }
        // Adds Liquid Glass to the button.
        .buttonStyle(.glass)
        #if os(macOS)
        .tint(.clear)
        #endif
        // Adds an identifier to the button for animation.
        .glassEffectID("togglebutton", in: namespace)
    }
    .frame(width: Constants.badgeFrameWidth)
}
```

See Also

App features

{} Landmarks: Applying a background extension effect

Configure an image to blur and extend under a sidebar or inspector panel.

{} Landmarks: Extending horizontal scrolling under a sidebar or inspector

Improve your horizontal scrollbar's appearance by extending it under a sidebar or inspector.

{} Landmarks: Refining the system provided Liquid Glass effect in toolbars

Organize toolbars into related groupings to improve their appearance and utility.