Windows

Introduction to SwiftUI

Hacking Spatial Computing • National University of Singapore

Ornaments

Creating Depth

Multi-Window Experiences

Ornaments

Creating Depth

Multi-Window Experiences

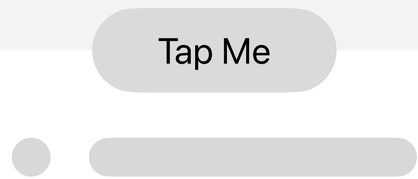
An ornament **presents controls and information** related to a window, **without crowding or obscuring the window's contents**.

```swift
import SwiftUI
import RealityKit
import RealityKitContent

struct ContentView: View {

    var body: some View {
        VStack {
            Model3D(named: "Scene", bundle: realityKitContentBundle)
                .padding(.bottom, 50)

            Text("Hello, world!")

            ToggleImmersiveSpaceButton()
        }
        .padding()
    }
}
```

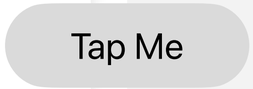
```swift
import SwiftUI
import RealityKit
import RealityKitContent

struct ContentView: View {

    var body: some View {
        VStack {
            Model3D(named: "Scene", bundle: realityKitContentBundle)
                .padding(.bottom, 50)

            Text("Hello, world!")

            ToggleImmersiveSpaceButton()
        }
        .padding()
        .ornament(attachmentAnchor: .parent(.bottom)) {
            Button("Tap Me!") {

            }
        }
    }
}
```


Hello, world!
Show Immersive Space
Tap Me

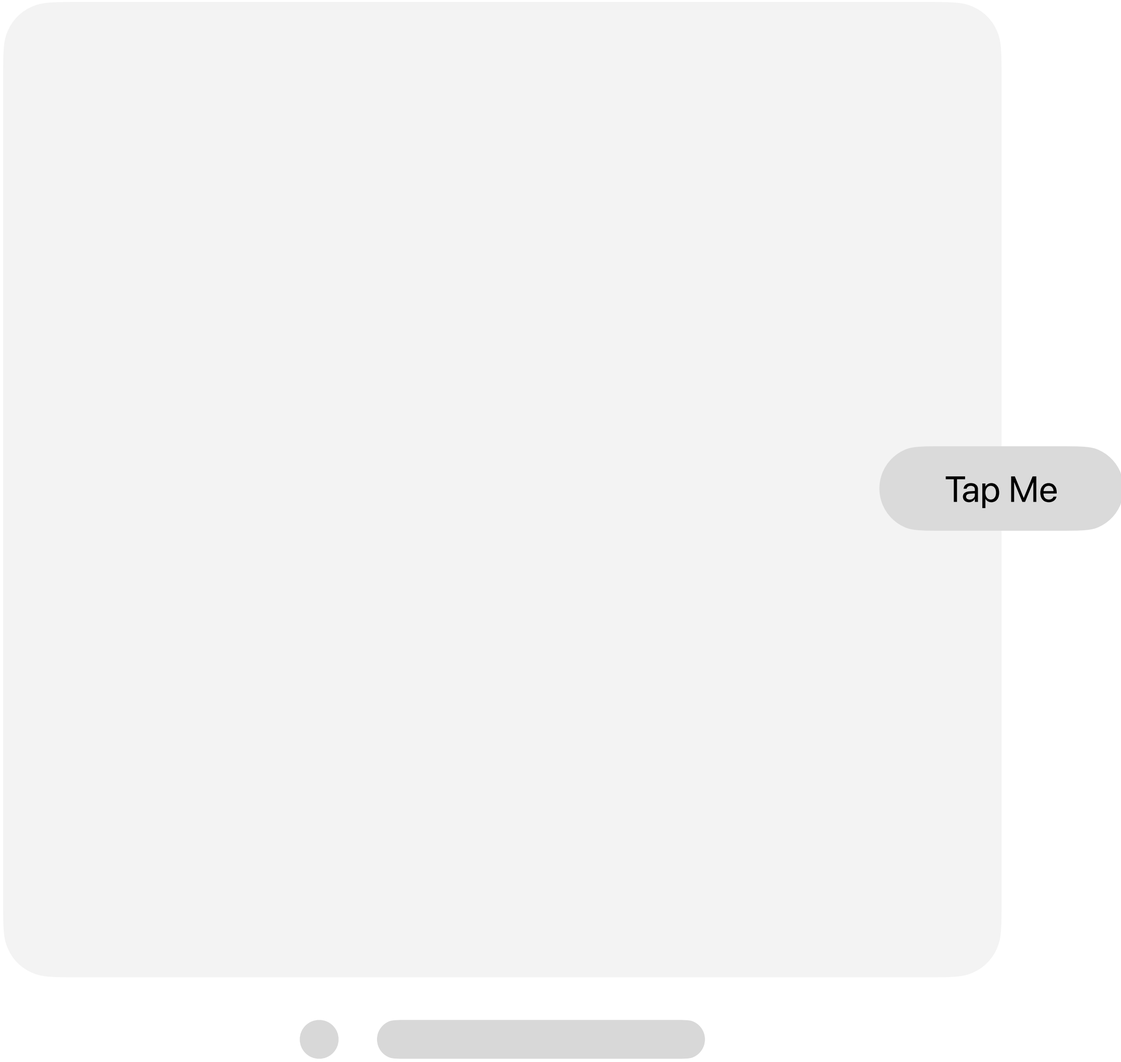
```
.ornament(attachmentAnchor:
        .parent(.bottom)) {
    Button("Tap Me!"){

    }
}
```



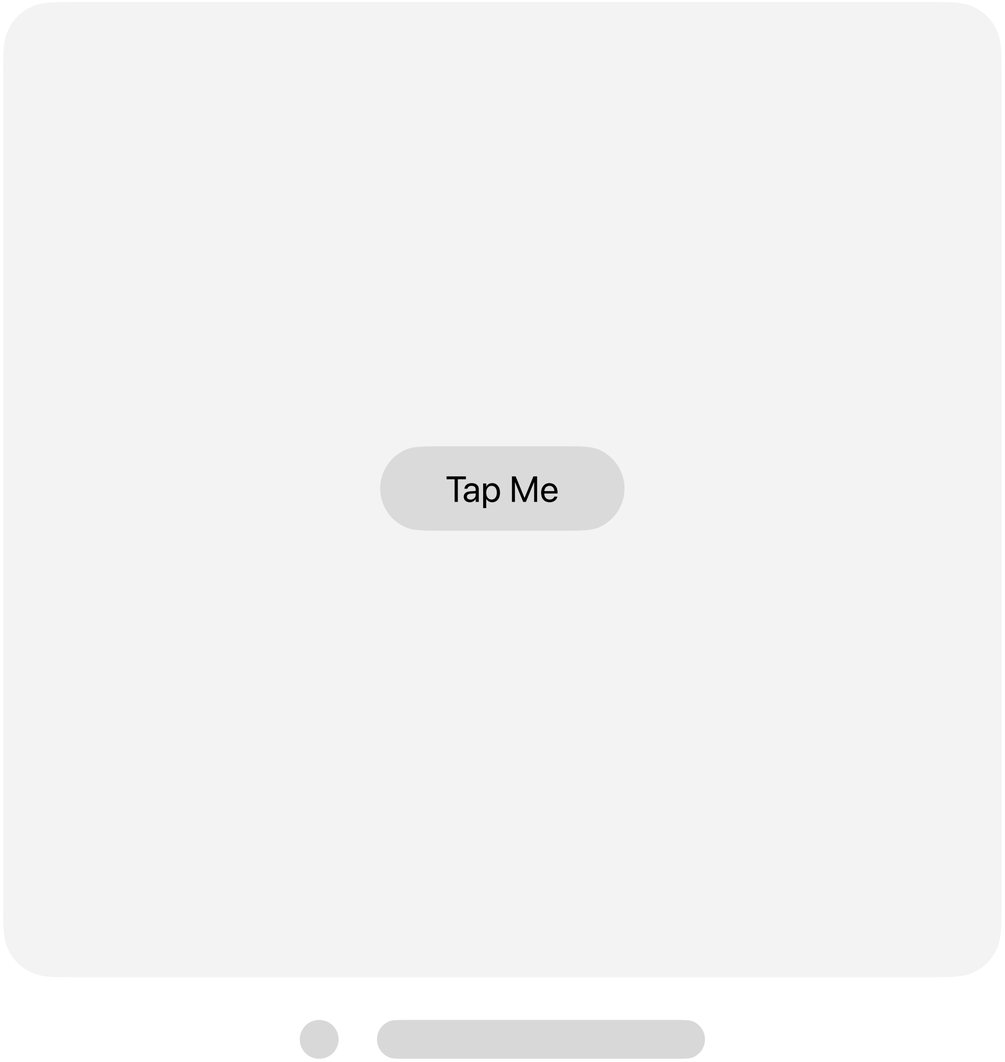
```
.ornament(attachmentAnchor:
        .parent(.leading)) {
    Button("Tap Me!"){

    }
}
```



```
.ornament(attachmentAnchor:
      .parent(.trailing)) {
    Button("Tap Me!"){

    }
}
```



```
.ornament(attachmentAnchor:
        .parent(.front)) {
    Button("Tap Me!"){

    }
}
```



```
.zero
.origin
.center
.bottom
.bottomBack
.bottomFront
.bottomLeading
.bottomLeadingBack
.bottomLeadingFront
.bottomTrailing
.bottomTrailingBack
.bottomTrailingFront
.leading
.leadingBack
.leadingFront
.top
.topBack
.topFront
.topLeading
.topLeadingBack
.topLeadingFront
.topTrailing
.topTrailingBack
.topTrailingFront
.trailing
.trailingBack
.trailingFront
.front
.back
```

Ornaments

Creating Depth

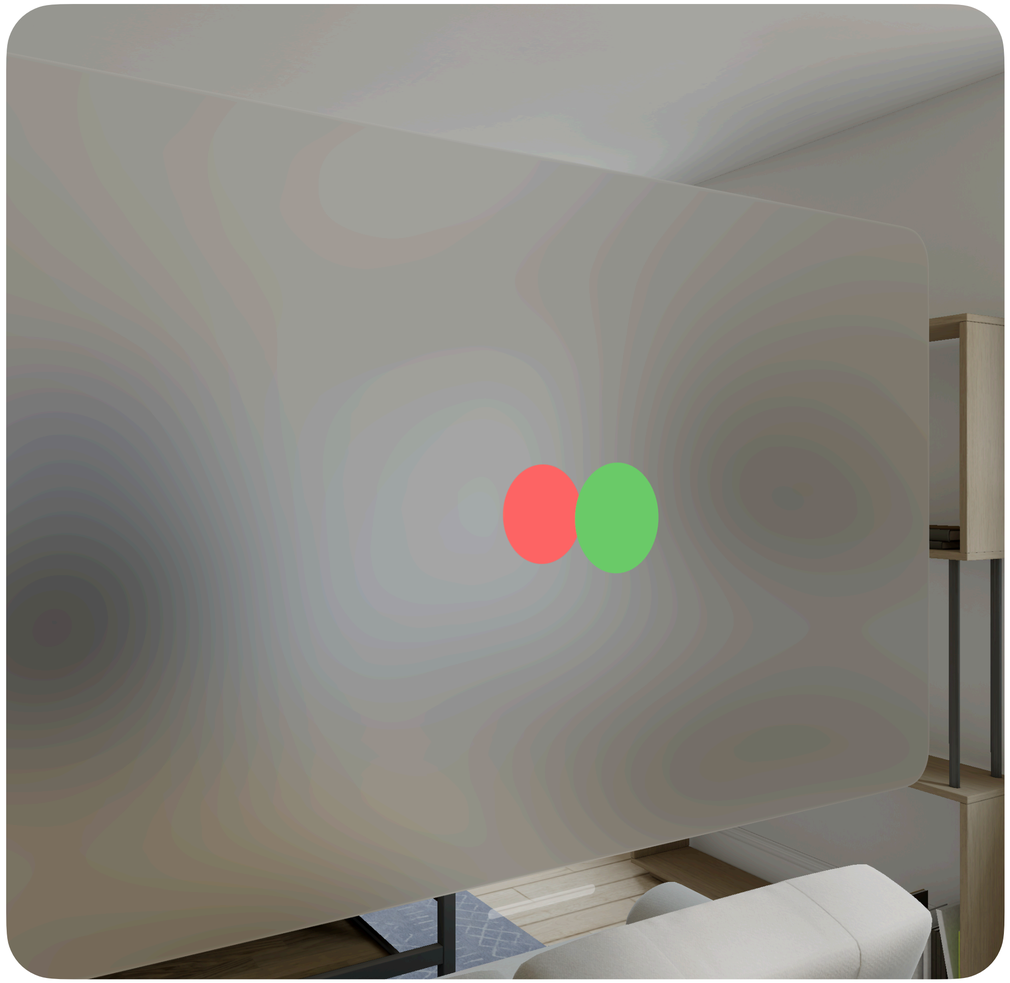
Multi-Window Experiences

On visionOS, ZStack allows you to create depth by **having views physically sit atop one another**.

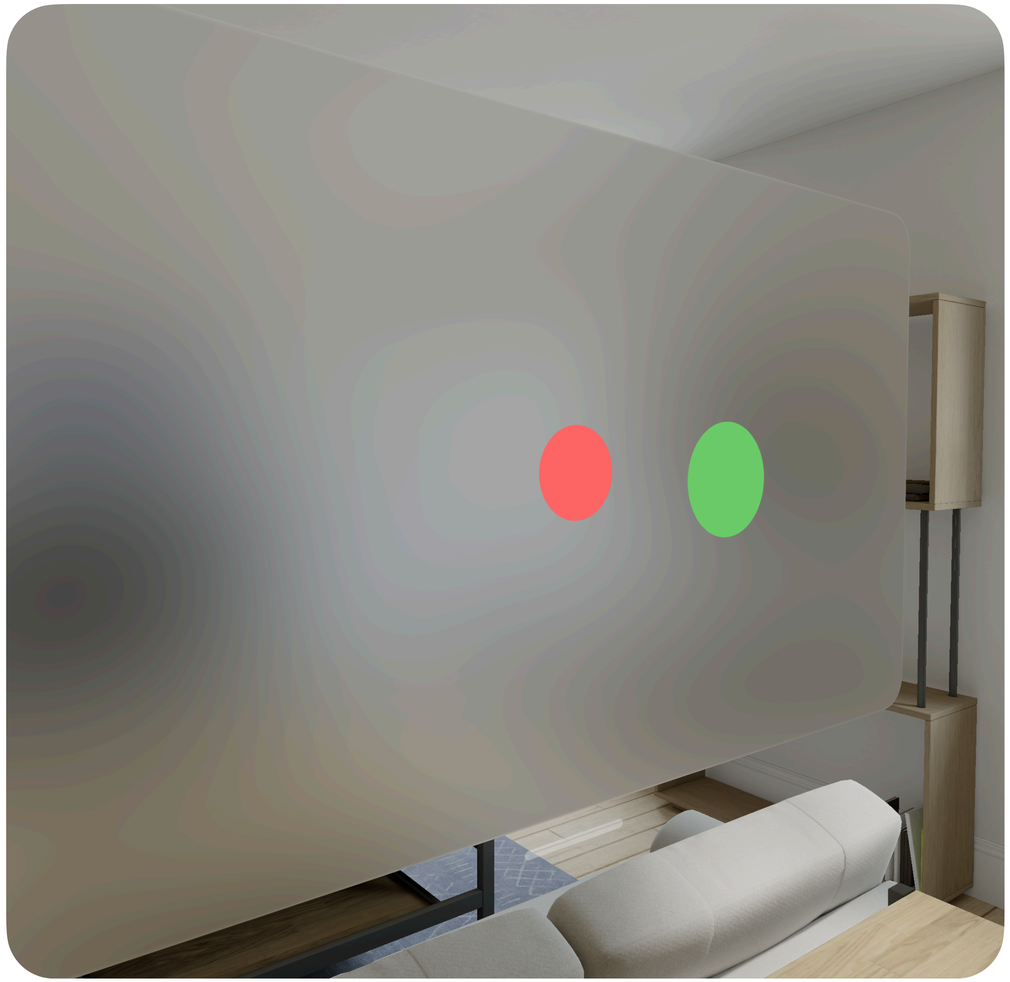
Padding 3D

Adds padding to a view in 3D space. Allowing you to **add space in front of and behind a view**.

```
ZStack {
    Circle()
        .fill(.red)
        .frame(width: 100, height: 100)
    Circle()
        .fill(.green)
        .frame(width: 100, height: 100)
        .padding3D(.back, 100)
}
```



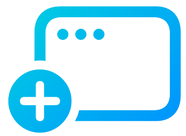
```
ZStack {
    Circle()
        .fill(.red)
        .frame(width: 100, height: 100)
    Circle()
        .fill(.green)
        .frame(width: 100, height: 100)
        .padding3D(.back, 200)
}
```



Ornaments

Creating Depth

Multi-Window Experiences



Creating Another Window

*AppName*App.swift **controls how your app is presented and its scenes**.

```swift
@main
struct WindowsVolumesApp: App {

    @State private var appModel = AppModel()

    var body: some Scene {
        WindowGroup {
            ContentView()
                .environment(appModel)
        }

        ImmersiveSpace(id: appModel.immersiveSpaceID) {
            ImmersiveView()
                .environment(appModel)
                .onAppear {
                    appModel.immersiveSpaceState = .open
                }
                .onDisappear {
                    appModel.immersiveSpaceState = .closed
                }
        }
        .immersionStyle(selection: .constant(.full), in: .full)
    }
}
```

WindowsVolumesApp.swift

```swift
@main
struct WindowsVolumesApp: App {

    @State private var appModel = AppModel()

    var body: some Scene {
        WindowGroup {
            ContentView()
                .environment(appModel)
        }

        WindowGroup(id: "SecondWindow") {
            Text("Hello from the second window!")
        }

        ImmersiveSpace(id: appModel.immersiveSpaceID) {
            ...
        }
        .immersionStyle(selection: .constant(.full), in: .full)
    }
}
```

WindowsVolumesApp.swift

Presenting a Window

```swift
@main
struct WindowsVolumesApp: App {

    @State private var appModel = AppModel()

    var body: some Scene {
        WindowGroup {
            ContentView()
                .environment(appModel)
        }

        ImmersiveSpace(id: appModel.immersiveSpaceID) {
            ImmersiveView()
                .environment(appModel)
                .onAppear {
                    appModel.immersiveSpaceState = .open
                }
                .onDisappear {
                    appModel.immersiveSpaceState = .closed
                }
        }
        .immersionStyle(selection: .constant(.full), in: .full)
    }
}
```

WindowsVolumesApp.swift

```swift
@main
struct WindowsVolumesApp: App {

    @State private var appModel = AppModel()

    var body: some Scene {
        WindowGroup {
            ContentView()
                .environment(appModel)
        }

        WindowGroup(id: "SecondWindow") {
            Text("Hello from the second window!")
        }

        ImmersiveSpace(id: appModel.immersiveSpaceID) {
            ...
        }
        .immersionStyle(selection: .constant(.full), in: .full)
    }
}
```

WindowsVolumesApp.swift

```swift
struct ContentView: View {

    @Environment(\.openWindow) private var openWindow

    var body: some View {
        Button("Show Window") {
            openWindow(id: "SecondWindow")
        }
    }
}
```

ContentView.swift

Passing Data

Binding variables can be used to pass data bidirectionally between windows as well.

```swift
@main
struct WindowsVolumesApp: App {

    @State private var appModel = AppModel()

    @State private var counter = 0

    var body: some Scene {
        WindowGroup {
            ContentView(counter: $counter)
                .environment(appModel)
        }

        WindowGroup(id: "SecondWindow") {
            Text("\(counter) clicks!")
        }

        ImmersiveSpace(id: appModel.immersiveSpaceID) {
            ...
        }
        .immersionStyle(selection: .constant(.full), in: .full)
    }
}
```

WindowsVolumesApp.swift

```swift
struct ContentView: View {

    @Environment(\.openWindow) private var openWindow

    @Binding var counter: Int

    var body: some View {
        VStack {
            Button("Show Window") {
                openWindow(id: "SecondWindow")
            }

            Button("Increment Counter") {
                counter += 1
            }
        }
    }
}
```

ContentView.swift

Activity

Show Even More Windows

- When the counter reaches 10, show a special pop-up window
- Feel free to be as creative as you want, show an image, an animation, whatever you want!

15-minute timer has not started.