

[visionOS](#) / [Introductory visionOS samples](#) / Creating 3D models as movable windows

Sample Code

Creating 3D models as movable windows

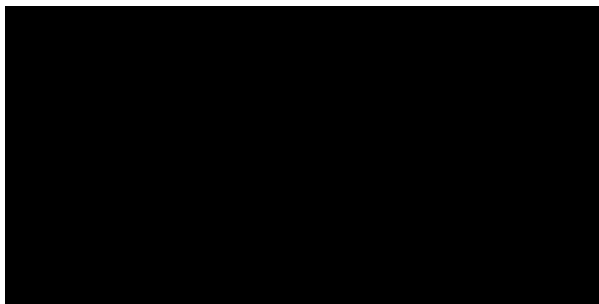
Display 3D content with a volumetric window that people can move.

Download

visionOS 2.0+ | Xcode 16.0+

Overview

This sample code project demonstrates how to create and display 3D models in a visionOS app, using volumetric windows that people can pick up and move around their environment. The app launches directly into a volumetric window containing a 3D model.



Play 

The volumetric window style in SwiftUI creates a 3D volumetric window that someone can pick up and move around their space, similar to a 2D window.

Set up volumetric window requirements

The `VolumetricWindow` view loads a USDZ file and displays its contents. The `defaultSize` value defines the dimensions of the initial volumetric window size.

```

import SwiftUI
import RealityKit

/// A view that loads in a 3D model and sets the dimensions of the volumetric window
struct VolumetricWindow: View {
    /// The default length of each side of the cubic volumetric window, in meters.
    static let defaultSize: CGFloat = 0.5

    // ...
}

```

The sample's `EntryPoint` structure defines a volumetric window group and a `defaultSize` value that sets the initial dimensions of the view. The `Info.plist` specifies the [UIApplicationPreferredDefaultSceneSessionRole](#) as `UIWindowSceneSessionRoleVolumetric` Application launching directly into the volumetric window.

```

import SwiftUI

@main
struct EntryPoint: App {
    /// The multiplier for the height of the volumetric window.
    let heightModifier: CGFloat = 0.25

    var body: some Scene {
        // Configure a window group with a volumetric window.
        WindowGroup() {
            VolumetricWindow()
        }
        .windowStyle(.volumetric)
        // Scale the size of the window group relative to the volumetric window's size
        .defaultSize(
            width: VolumetricWindow.defaultSize,
            height: heightModifier * VolumetricWindow.defaultSize,
            depth: VolumetricWindow.defaultSize,
            in: .meters
        )
    }
}

```

SwiftUI requires three separate arguments to define the width, height, and depth of the volumetric window.

Load a 3D model as an entity in a view

The `VolumetricWindow` view loads a USDZ file as an `Entity` instance and adds it to the scene.

```
RealityView { content in
  // Attempt to load the entity that uses the filename as a source.
  guard let model = try? await ModelEntity(named: modelName) else {
    return
  }

  // Add the model to the `RealityView`.
  content.add(model)
}
```

Adjust the entity to the size of the volumetric window

The sample updates the scale and position of the entity in the update closure of `RealityView` to respond to volume resizing.

The `viewBounds` object converts the `GeometryReader3D` coordinate space to a local-world coordinate space that `RealityView` uses.

```
let viewBounds = content.convert(
  geometry.frame(in: .local),
  from: .local,
  to: .scene
)
```

The `visualBounds(recursive:relativeTo:excludeInactive:)` method computes a bounding box that contains the outer dimensions of the entity.

The view bounding box places the model at the bottom of the y-axis value of the window's view:

```
// Set the model's position to the bottom of the visual bounding box.
model.position.y -= model.visualBounds(relativeTo: nil).min.y

// Adjust the model's position on the y-axis to align with the view bounds.
model.position.y += viewBounds.min.y
```

The app sets the scale of the model to match the width of the view bounds.

```
/// The base size of the model when the scale is 1.
let baseExtents = model.visualBounds(relativeTo: nil).extents / model.scale

/// The scale required for the model to fit the bounds of the volumetric window.
let scale = Float(viewBounds.extents.x) / baseExtents.x

// Apply the scale to the model to fill the full size of the window.
model.scale = SIMD3<Float>(repeating: scale)
```

See Also

Related samples



Creating SwiftUI windows in visionOS

Display and manage multiple SwiftUI windows in your visionOS app.