Sample Code

# Incorporating real-world surroundings in an immersive experience

Create an immersive experience by making your app's content respond to the local shape of the world.

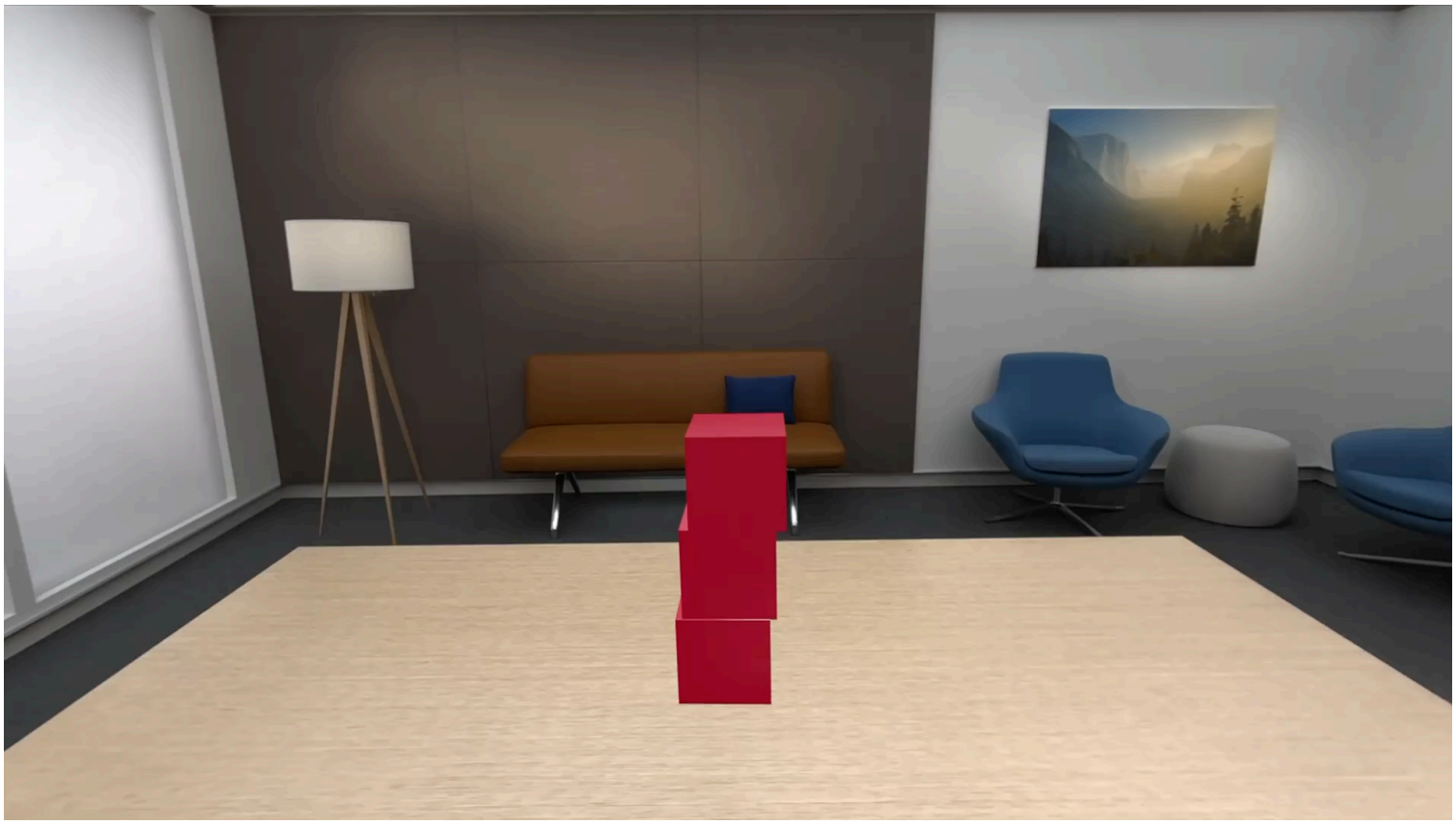
Download

visionOS 1.0+ | Xcode 15.1+

## Overview

Scene reconstruction helps bridge the gap between the rendered 3D content in your app and the person's surroundings. Use scene reconstruction in ARKit to give your app an idea of the shape of the person's surroundings and to bring your app experience into their world. Immersive experiences — those that use the `mixed` space style — are best positioned to incorporate this kind of contextual information: scene reconstruction is only available in spaces and isn't as relevant for the `full` space style.

In addition to providing a 3D mesh of the shape of different nearby objects, ARKit gives a classification to each mesh face it detects. For example, it might classify a face of a mesh as being part of an appliance, a piece of furniture, or structural information about the room like the position of walls and floors. The following video shows virtual cubes colliding with the scene reconstruction mesh, which makes the cubes appear to land on a table:



Play ⊙

## Configure a scene reconstruction session

Scene reconstruction requires the `ARKitSession.AuthorizationType.worldSensing` authorization type and corresponding usage description that you supply in your app's `Info.plist` file. The following starts a session and processes updates as ARKit refines its reconstruction of the person's surroundings:

```
RealityView { content in
    content.add(model.setupContentEntity())
}
.task {
    do {
        if model.dataProvidersAreSupported && model.isReadyToRun {
            try await model.session.run([model.sceneReconstruction, model.handTracki
        } else {
            await dismissImmersiveSpace()
        }
    } catch {
        logger.error("Failed to start session: \(error)")
        await dismissImmersiveSpace()
        openWindow(id: "error")
    }
}
```

```
.task {
    await model.processHandUpdates()
}
.task {
    await model.monitorSessionEvents()
}
.task(priority: .low) {
    await model.processReconstructionUpdates()
}
.gesture(SpatialTapGesture().targetedToAnyEntity().onEnded { value in
    let location3D = value.convert(value.location3D, from: .global, to: .scene)
    model.addCube(tapLocation: location3D)
})
```

## Add real-world interactivity using collision components

You can make rendered 3D content more lifelike by having it appear to interact physically with objects in the person's surroundings, like furniture and floors. Use RealityKit's collision components and physics support to provide these interactions in your app. The `generateStaticMesh(from:)` method bridges between scene reconstruction and RealityKit's physics simulation.

**Warning**

Be mindful of how much content you include in immersive scenes that use the `mixed` style. Content that fills a significant portion of the screen, even if that content is partially transparent, can prevent the person from seeing potential hazards in their surroundings. If you want to immerse the person in your content, configure your space with the `full` style. For more information, see Creating fully immersive experiences in your app.

Use low-priority tasks to generate meshes, because generating them is a computationally expensive operation. The following creates a mesh entity with collision shapes using scene reconstruction:

```
func processReconstructionUpdates() async {
    for await update in sceneReconstruction.anchorUpdates {
        let meshAnchor = update.anchor

        guard let shape = try? await ShapeResource.generateStaticMesh(from: meshAnch
        switch update.event {
        case .added:
```

```
            let entity = ModelEntity()
            entity.transform = Transform(matrix: meshAnchor.originFromAnchorTransfor
            entity.collision = CollisionComponent(shapes: [shape], isStatic: true)
            entity.components.set(InputTargetComponent())

            entity.physicsBody = PhysicsBodyComponent(mode: .static)

            meshEntities[meshAnchor.id] = entity
            contentEntity.addChild(entity)
        case .updated:
            guard let entity = meshEntities[meshAnchor.id] else { continue }
            entity.transform = Transform(matrix: meshAnchor.originFromAnchorTransfor
            entity.collision?.shapes = [shape]
        case .removed:
            meshEntities[meshAnchor.id]?.removeFromParent()
            meshEntities.removeValue(forKey: meshAnchor.id)
        }
    }
}
```

> **Note**
>
> Scene reconstruction meshes only support the `PhysicsBodyMode.static` physics body component mode.

Each object in the scene reconstruction mesh updates its `originFromAnchorTransform` information independently and requires a separate static mesh because ARKit subdivides its representation of the world into multiple, distinct sections.

## Display scene reconstruction meshes during debugging

People using an app that leverages scene reconstruction typically don't need to see a visual rendering of the scene reconstruction mesh. The system already shows passthrough video in an immersive experience. However, temporarily displaying the scene reconstruction mesh can help while you're developing and debugging your app. In Xcode's debugging toolbar, click the Enable Visualizations button and select Collision Shapes. Because each element of the scene reconstruction mesh has a collision component, the details of the mesh appear in the debug visualization. For more information, see Diagnosing issues in the appearance of a running app.

## See Also

## ARKit

{} Happy Beam

Leverage a Full Space to create a fun game using ARKit.

Setting up access to ARKit data

Check whether your app can use ARKit and respect people's privacy.

{} Placing content on detected planes

Detect horizontal surfaces like tables and floors, as well as vertical planes like walls and doors.

{} Tracking specific points in world space

Retrieve the position and orientation of anchors your app stores in ARKit.

Tracking preregistered images in 3D space

Place content based on the current position of a known image in a person's surroundings.

{} Exploring object tracking with ARKit

Find and track real-world objects in visionOS using reference objects trained with Create ML.

{} Object tracking with Reality Composer Pro experiences

Use object tracking in visionOS to attach digital content to real objects to create engaging experiences.

{} Building local experiences with room tracking

Use room tracking in visionOS to provide custom interactions with physical spaces.

{} Placing entities using head and device transform

Query and react to changes in the position and rotation of Apple Vision Pro.