Sample Code

Creating 3D entities with RealityKit

Display a horizontal row of three-dimensional shapes in your visionOS app, using predefined mesh and white material.

Download

visionOS 2.0+ | Xcode 16.0+

Overview

This sample code project demonstrates how to create and display 3D objects using predefined methods in `MeshResource`, such as `generateBox(size:cornerRadius:)` for creating a box and `generateSphere(radius:)` for creating a sphere. Additionally, it shows how to use `SimpleMaterial` to apply a white material. By combining the mesh and material, you can create a `ModelComponent` for your visionOS app with RealityKit.

The sample app draws the following entities in its main view:

- A box
- A rounded box
- A sphere
- A cone
- A cylinder

RealityKit defines the `Box`, `Sphere`, `Cone`, and `Cylinder` types, and the app creates a `rounded Box` type by adding a `cornerRadius` to `Box`.

Add 3D shapes to the view

The app's main view renders the entity using `RealityView` and adds the shape using the `addGeometryShapes(to:)` method.

```
import SwiftUI
import RealityKit

struct ShapesView: View {
    var body: some View {
        RealityView { content in
            addGeometryShapes(to: content)
        }
    }
```

```
    // Positions and adds all geometric shapes to the scene.
    func addGeometryShapes(to content: RealityViewContent) {
        /// An array of all the 3D entities representing different shapes.
        let allGeometryEntities = [
            ShapesView.boxEntity,
            ShapesView.roundedBoxEntity,
            ShapesView.sphereEntity,
            ShapesView.coneEntity,
            ShapesView.cylinderEntity
        ]

        var xOffset: Float = -0.25

        // Position the entities along the x-axis, and add them to the content.
        for entity in allGeometryEntities {
            entity.position.x = xOffset
            content.add(entity)
            xOffset += 0.125
        }
    }
}
```

The `addGeometryShapes(to:)` method sets an array of 3D entities in a row along the x-axis, including a box, a rounded box, a sphere, a cone, and a cylinder, with an offset of 0.125 between each.

Create a primitives 3D view extension

The `ShapesView` extension defines the entity types for the box, sphere, cone, and cylinder with white material by:

- Creating a material with `SimpleMaterial`
- Generating a mesh with `MeshResource`
- Rendering a 3D model with `ModelComponent`

```
import RealityKit

extension ShapesView {
    /// The white material that responds to lighting.
    static let whiteMaterial = SimpleMaterial(color: .white, isMetallic: false)
```

```
    /// The entity with a box geometry.
    static let boxEntity: Entity = {
        // Create an entity instance.
        let entity = Entity()

        // Create a mesh resource.
        let boxSize: Float = 0.1
        let boxMesh = MeshResource.generateBox(size: boxSize)

        // Add the mesh resource to a model component, and add it to the entity.
        entity.components.set(ModelComponent(mesh: boxMesh, materials: [whiteMateria

        return entity
    }()

    // ...
}
```

In this sample code project, the `boxEntity` property creates an entity and generates a box mesh with a length of 0.1 units. Use the mesh and the white material to create a `ModelComponent`, and then set it as a component and return the entity.

Similarly, you can create 3D shapes using other preset meshes, such as those in the example below (a sphere, cone, and cylinder):

```
import RealityKit

extension Primitives3DView {
    // ...

    /// The entity with a spherical geometry.
    static let sphereEntity: Entity = {
        // Create an entity instance.
        let entity = Entity()

        // Create a mesh resource.
        let sphereRadius: Float = 0.05
        let boxMesh = MeshResource.generateSphere(radius: sphereRadius)

        // Add the mesh resource to a model component, and add it to the entity.
        entity.components.set(ModelComponent(mesh: boxMesh, materials: [whiteMateria

        return entity
```

```
    }()

    /// The entity with a conical geometry.
    static let coneEntity: Entity = {
        // Create an entity instance.
        let entity = Entity()

        // Create a mesh resource.
        let coneHeight: Float = 0.1
        let coneRadius: Float = 0.05
        let roundedBoxMesh = MeshResource.generateCone(height: coneHeight, radius: c

        // Add the mesh resource to a model component, and add it to the entity.
        entity.components.set(ModelComponent(mesh: roundedBoxMesh, materials: [white

        return entity
    }()

    /// The entity with a cylindrical geometry.
    static let cylinderEntity: Entity = {
        // Create an entity instance.
        let entity = Entity()

        // Create a mesh resource.
        let cylinderHeight: Float = 0.1
        let cylinderRadius: Float = 0.05
        let roundedBoxMesh = MeshResource.generateCylinder(height: cylinderHeight, r

        // Add the mesh resource to a model component, and add it to the entity.
        entity.components.set(ModelComponent(mesh: roundedBoxMesh, materials: [white

        return entity
    }()
}
```

See Also

Related samples

{} Creating 2D shapes with SwiftUI

Draw two-dimensional shapes in your visionOS app with SwiftUI shapes or with your custom shapes.

{} Combining 2D and 3D views in an immersive app

Use attachments to place 2D content relative to 3D content in your visionOS app.

Related articles

Understanding the modular architecture of RealityKit

Learn how everything fits together in RealityKit.