Sample Code

Loading entities with ShaderGraph materials

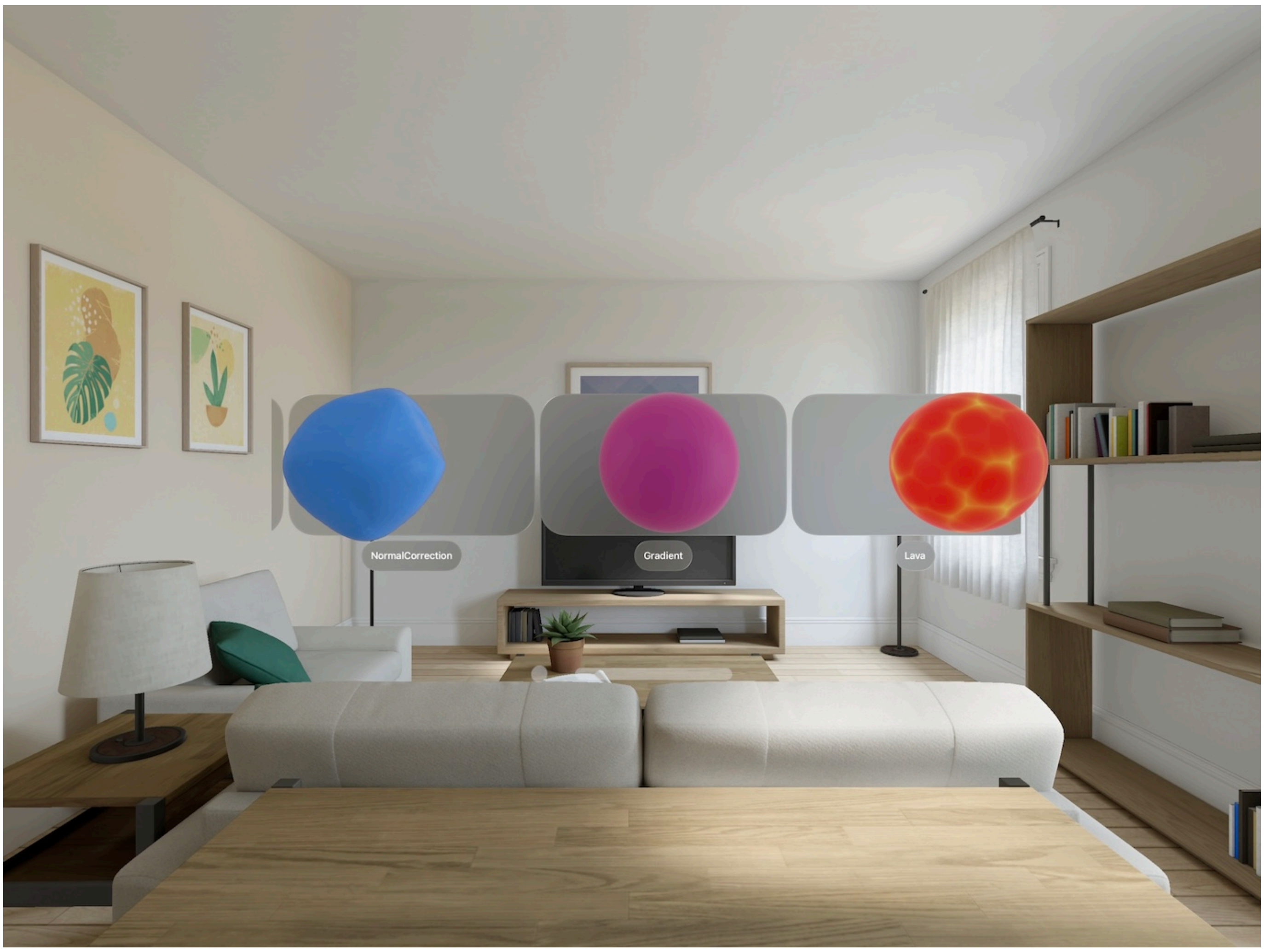
Bring entities that contain materials created with Reality Composer Pro for use in your visionOS app.

Download

visionOS 2.0+ | Xcode 16.2+

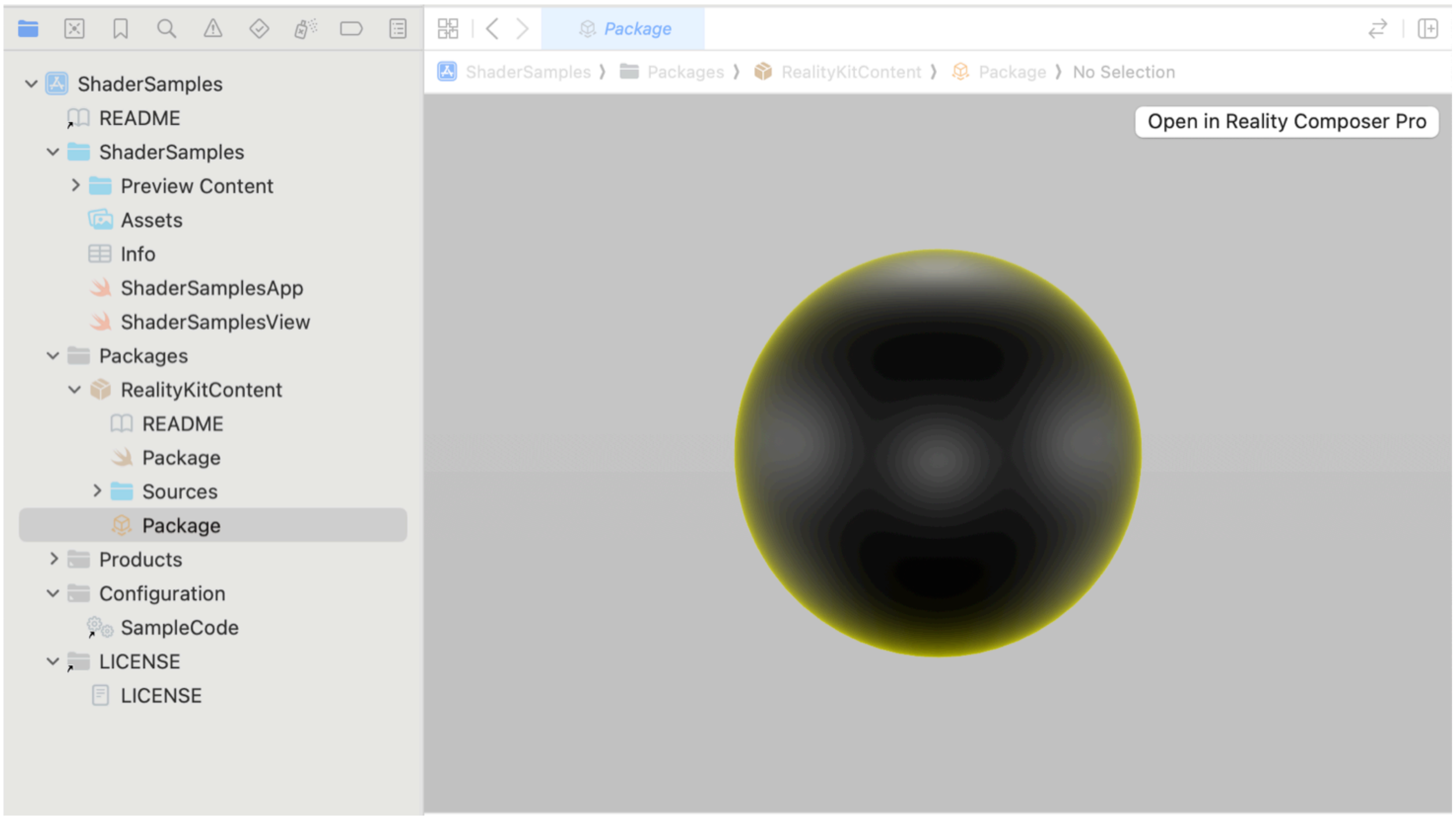
Overview

Reality Composer Pro gives you the ability to create entities that you can bring over to, and interact with, in your visionOS app. This sample loads the entities created with `ShaderGraph` and places them in a `ScrollView`.

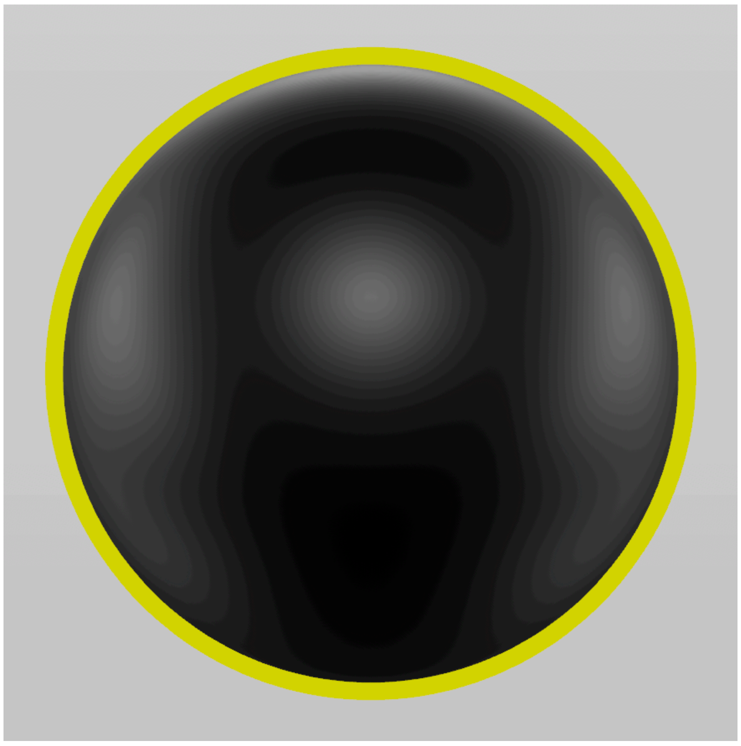


Create the materials in ShaderGraph

In the Project Navigator, open Packages > RealityKitContent, select `Package.realitycomposerpro`, then click Open in Reality Composer Pro.

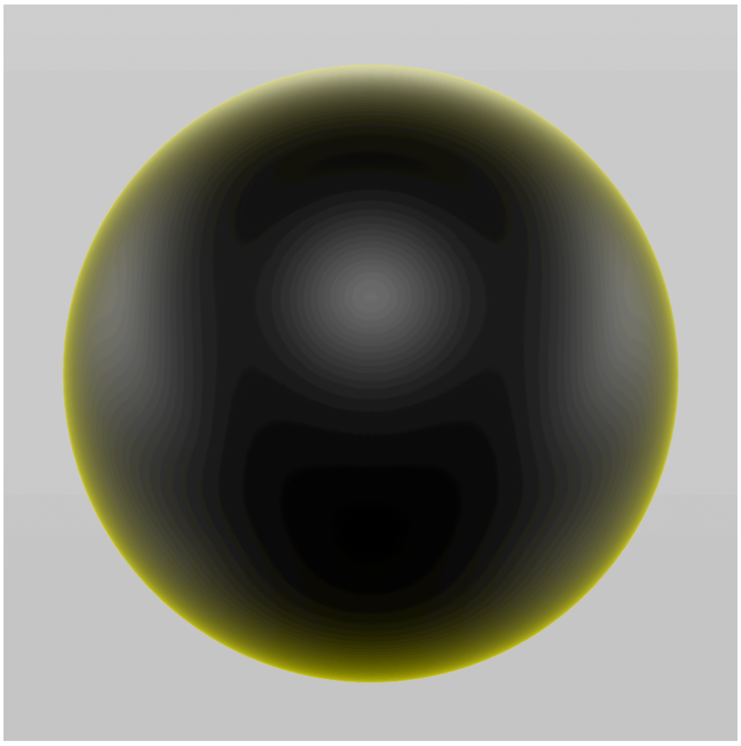


The `RealityKitContent` package in the sample includes these shaders:

Shader	**Shader Preview**
Outline shader In Reality Composer Pro, open the `Outline Shader.usda` file and click on `/Root/OutlineShaderEntity/Sphere Outline/OutlineShaderMaterial` to see the material in ShaderGraph.	

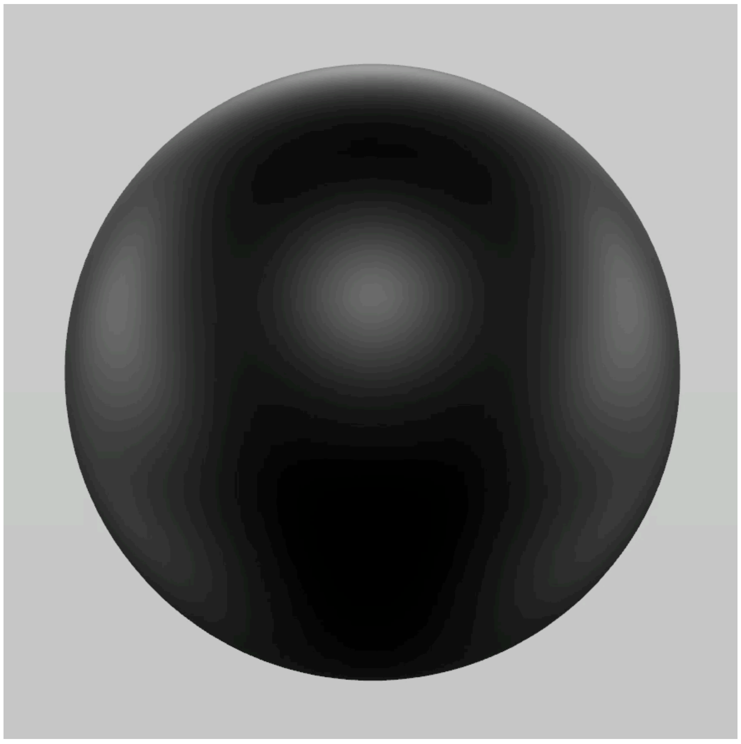
Fresnel shader

In Reality Composer Pro, open the `Fresnel Shader.usda` file and click on `/Root/FresnelMaterial` to see the material in ShaderGraph (for more information, see Creating a Fresnel outline effect with Shader Graph).



Dissolve shader

In Reality Composer Pro, open the `Dissolve Shader.usda` file and click on `/Root/DissolveMaterial` to see the material in ShaderGraph.



Play ⊙

Vertex displacement shader

In Reality Composer Pro, open the `Vertex DisplacementShader.usda` file and click on `/Root/VertexDisplacementMaterial` to see the material in ShaderGraph (for more information, see Creating a vertex displacement material with Shader Graph).

Play ⊙

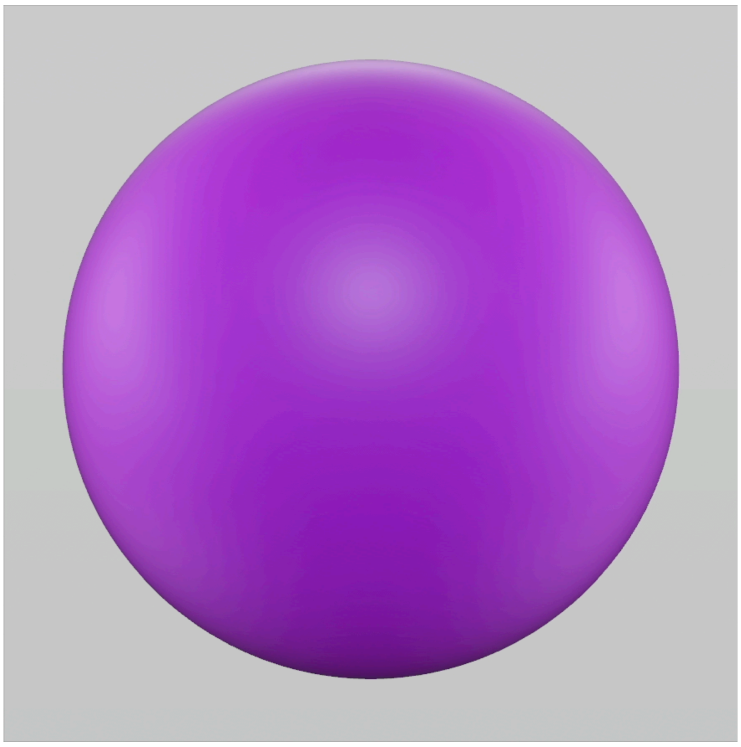
Vertex displacement shader with corrected normals

In Reality Composer Pro, open the `Normal CorrectionShader.usda` file and click on `/Root/NormalCorrectionMaterial` to see the material in ShaderGraph (for more information, see Correcting normals after vertex displacement with Shader Graph).

Play ⊙

Gradient shader

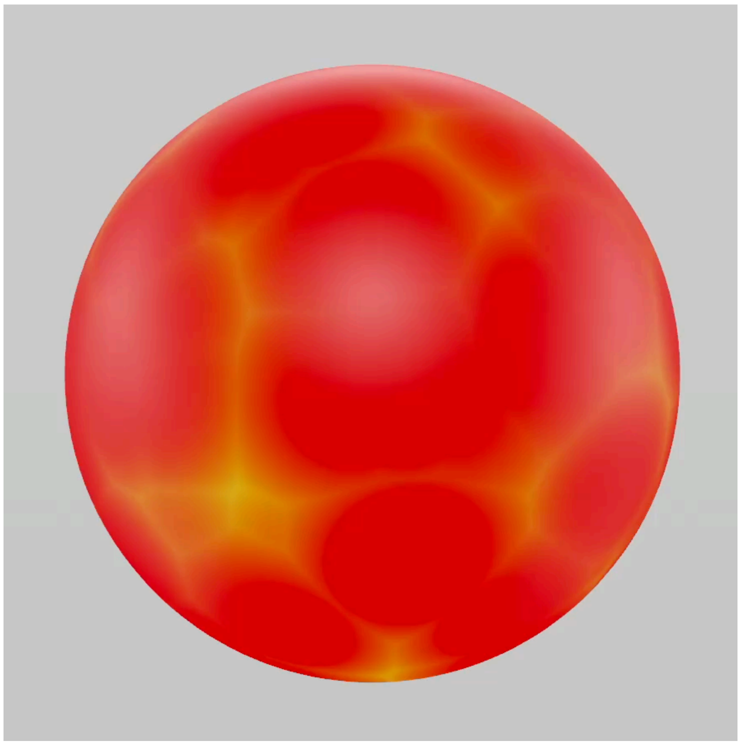
In Reality Composer Pro, open the `Gradient Shader.usda` file and click on `/Root/GradientMaterial` to see the material in ShaderGraph.



Play ⦿

Lava shader

In Reality Composer Pro, open the `Lava Shader.usda` file and click on `/Root/Lava Material` to see the material in ShaderGraph.



Play ⦿

Simple toon shader

In Reality Composer Pro, open the `Toon Shader.usda` file and click on `/Root/Toon Material` to see the material in ShaderGraph.

Physically Based Rendering (PBR) toon shader

In Reality Composer Pro, open the `PBRToon Shader.usda` file and click on `/Root/Transform/Toy Biplane/geom/realistic/materials/usdpreviewsurface3sg` to see the material in ShaderGraph.

Create a volumetric view in the app

The sample uses a volumetric view in a `WindowGroup` to load the entities from the `RealityKit Content` package:

```
struct ShaderSamplesApp: App {
    var body: some Scene {
        WindowGroup {
            ShaderSamplesView()
        }
        .windowStyle(.volumetric)
    }
}
```

Load the shaders from the content package

The `loadShaders` method loads all the shader entities containing the shader materials from the `ShaderSamplesScene` scene in the `realityKitContentBundle`:

```swift
private func loadShaders() async {
    guard
        let shaderSamplesSceneRoot = try? await Entity(
            named: "ShaderSamplesScene", in: realityKitContentBundle
        ).children.first
    else {
        return
    }

    // Get the shader sample entities.
    for shaderSampleEntity in shaderSamplesSceneRoot.children {
        shaderSampleEntities.append(shaderSampleEntity)
    }
}
```

The method saves the entities in a list named `shaderSampleEntities` to be used later to create the `RealityViewContent`.

Add the shader entities into the RealityView content

The `createRealityView` method in the sample creates a `RealityView` for an entity and adds it to the `RealityViewContent` in the `createRealityView` method:

```swift
private func createRealityView(_ shaderSampleEntity: Entity) -> some View {
    RealityView { content in
        shaderSampleEntity.position = [0, 0, 0]
        shaderSampleEntity.scale = SIMD3<Float>(
            repeating: 0.75)
        content.add(shaderSampleEntity)
    }
    .glassBackgroundEffect()
    .containerRelativeFrame(
        [.horizontal], count: 3, spacing: 5)
}
```

The method positions the `shaderSampleEntity` at the center of the `RealityView` and reduces the scale to fit in the `ScrollView`.

Load the entities into a ScrollView

The sample uses a `ScrollView` to display all the entities in the `shaderSampleEntities` list in a `RealityView`. The `ScrollView` shows the name of the shader below the `RealityView`:

```
var body: some View {
    ScrollView(.horizontal) {
        HStack {
            ForEach(shaderSampleEntities) { shaderSampleEntity in
                VStack {
                    // Display the shader sample entity.
                    createRealityView(shaderSampleEntity)

                    // Display the shader name.
                    createDisplayName(shaderSampleEntity)
                }.padding([.bottom])
            }
        }
    }.task {
        // Load the shader samples scene root.
        await loadShaders()
    }
}
```

The sample loads the shaders in an asynchronous `Task`, and displays the shaders in `Scroll View`.

Fresnel
Dissolve
VertexDisplacement

Play

See Also

Shader Graph

Designing materials with Shader Graph

Create realistic materials with Shader Graph's node editor in Reality Composer Pro.

Creating a Fresnel outline effect with Shader Graph

Highlight a model by adding an outline effect.

Creating a vertex displacement material with Shader Graph

Animate the vertices of a mesh over time with 3D Perlin noise by creating a Shader Graph material.

Correcting normals after vertex displacement with Shader Graph

Approximate new vertex normals for materials that displace a mesh's vertices.