Structure

# HoverEffectComponent

**A component that applies a visual effect to a hierarchy of entities when a person looks at or selects an entity.**

iOS 18.0+ | iPadOS 18.0+ | Mac Catalyst 18.0+ | macOS 15.0+ | visionOS 1.0+

```
struct HoverEffectComponent
```

## Overview

Add a HoverEffectComponent to an entity to convey the entity can receive gestures or to separate parts that can be interacted with independently.

> **Note**
>
> On visionOS, you hover over an entity by looking at or directly touching it. On other platforms, you hover over an entity by moving your mouse cursor over it.

> **Important**
>
> Entities also need to have an `InputTargetComponent` and `CollisionComponent` to receive hover effects.

```
let boxSize = SIMD3<Float>(0.5, 0.1, 0.05)

let modelComponent = ModelComponent(
    mesh: MeshResource.generateBox(size: boxSize),
    materials: [SimpleMaterial(color: .black, roughness: 0.5, isMetallic: false)]
```

```
)
let collisionComponent = CollisionComponent(
    shapes: [ShapeResource.generateBox(size: boxSize)]
)
let inputTargetComponent = InputTargetComponent()
let hoverEffectComponent = HoverEffectComponent()

let entityA = Entity()
entityA.components.set([modelComponent, collisionComponent, inputTargetComponent, ho
```

The example code above produces the following scene.

Play ⊙

## Hierarchical behavior

The `HoverEffectComponent` applies its effect to an entity's entire hierarchy, which means that all the entity's descendent entities inherit the effect when a person looks at (or selects) an entity.

```
let boxSize = SIMD3<Float>(0.5, 0.1, 0.05)

// Create a black box model, a matching collision component, and an input target com
let modelComponent = ModelComponent(
    mesh: MeshResource.generateBox(size: boxSize),
    materials: [SimpleMaterial(color: .black, roughness: 0.5, isMetallic: false)]
)
let collisionComponent = CollisionComponent(
    shapes: [ShapeResource.generateBox(size: boxSize)]
)
let inputTargetComponent = InputTargetComponent()

// Create a default hover effect component.
let hoverEffectComponent = HoverEffectComponent()

let entityA = Entity()
```

```
entityA.components.set([modelComponent, collisionComponent, inputTargetComponent])

let entityB = entityA.clone(recursive: true)
let entityC = entityB.clone(recursive: true)

// B is a child of A.
entityA.addChild(entityB)
// C is a child of B.
entityB.addChild(entityC)

// Place B above its parent, A.
entityB.position.y = 0.1
// Place C above its parent, B.
entityC.position.y = 0.1

entityA.components.set(hoverEffectComponent)
```

In this example only the highest entity in the hierarchy has a `HoverEffectComponent`. Activating this hover effect causes all of its children to display the same effect.

Play ⊙

## Grouping multiple hover effects

The optional `groupID` connects entities so that they activate their hover effects together, which is independent from the entities' hierarchy. When someone hovers over any entity in the group, all entities with the same `HoverEffectComponent.GroupID` instance start their hover effects together.

```
let boxSize = SIMD3<Float>(0.5, 0.1, 0.05)
let modelComponent = ModelComponent(
    mesh: MeshResource.generateBox(size: boxSize, cornerRadius: 0.2),
    materials: [SimpleMaterial(color: .black, roughness: 0.5, isMetallic: false)]
)
let collisionComponent = CollisionComponent(
    shapes: [ShapeResource.generateBox(size: boxSize)]
```

```
)
let inputTargetComponent = InputTargetComponent()

let hoverEntityA = Entity(components:
    modelComponent, collisionComponent, inputTargetComponent
)
let hoverEntityB = hoverEntityA.clone(recursive: true)
hoverEntityB.position.y = -0.1
let hoverEntityC = hoverEntityA.clone(recursive: true)
hoverEntityC.position.y = -0.2

content.add(hoverEntityA)
content.add(hoverEntityB)
content.add(hoverEntityC)

// Create hover effect components for the two topmost entities in the scene.
var hoverA = HoverEffectComponent(
    .highlight(HoverEffectComponent.HighlightHoverEffectStyle(
            color: .green, strength: 2.0
        )))

var hoverB = HoverEffectComponent(
    .highlight(HoverEffectComponent.HighlightHoverEffectStyle(
            color: .yellow, strength: 2.0
        )))

// Create a `HoverEffectComponent.GroupID` instance and assign it to the hover effec
let topGroupID = HoverEffectComponent.GroupID()
hoverA.hoverEffect.groupID = topGroupID
hoverB.hoverEffect.groupID = topGroupID
hoverEntityA.components.set(hoverA)
hoverEntityB.components.set(hoverB)

// Create hover effect component for the bottommost entity in the scnee.
var hoverC = HoverEffectComponent(
    .highlight(HoverEffectComponent.HighlightHoverEffectStyle(
            color: .blue, strength: 2.0
        )))

// Create a `HoverEffectComponent.GroupID` instance and assign it to the hover effec
let bottomGroupID = HoverEffectComponent.GroupID()
hoverC.hoverEffect.groupID = bottomGroupID
hoverEntityC.components.set(hoverC)
```

In this example, the top and middle entities, `entityA` and `entityB`, have a `HoverEffectComponent` with the same group identifier. However, the bottom entity, `entityC`, has a different group ID. The top amd middle entity both glow when a person activates either one. The bottom entity glows separately and only when someone activates it directly.

Play ⊙

## Styles

You can customize the visual effect of the `HoverEffectComponent` through the use of styles.

The `HoverEffectComponent.SpotlightHoverEffectStyle` represents a feathered effect that follows the hover location on the entity. This is the default effect RealityKit uses if you create a `HoverEffectComponent` with the default initializer, `init()`. You can customize the color and strength of this effect by passing values into the `HoverEffectComponent.SpotlightHoverEffectStyle` constructor.

```
let hoverComponent = HoverEffectComponent(.spotlight(
    HoverEffectComponent.SpotlightHoverEffectStyle(
        color: .green, strength: 2.0
    )
))
```

Play

The `HoverEffectComponent.HighlightHoverEffectStyle` applies a uniform glow to the entity in addition to the feathered effect from the `HoverEffectComponent.SpotlightHoverEffectStyle`. You can customize the color and strength of this effect by passing values into the `HoverEffectComponent.HighlightHoverEffectStyle` constructor.

```
let hoverComponent = HoverEffectComponent(.highlight(
    HoverEffectComponent.HighlightHoverEffectStyle(
        color: .green, strength: 2.0
    )
))
```

Play

## Opacity functions

Customize how a `HoverEffectComponent` presents on a partially or fully transparent material with the following opacity functions:

`HoverEffectComponent.OpacityFunction.full`
: An opacity function that causes the effect to draw at full opacity regardless of opacity of the base material. This is the default opacity function.

`HoverEffectComponent.OpacityFunction.mask`
: An opacity function that draws the hover effect at full opacity when the entity's base material opacity is greater than `0.05`, and fades out the hover effect with the entity's base material when its opacity is equal to or less than `0.05`.

`HoverEffectComponent.OpacityFunction.blend`

An opacity function that draws the hover effect with an opacity that's equal to the product of the entity's base material and the shader's output.

`.full` `.mask` `.blend`

Play ⊙ Play ⊙ Play ⊙

> **Note**
>
> Opacity functions can't be applied to `HoverEffectComponents` that use `HoverEffectComponent.ShaderHoverEffectInputs`.

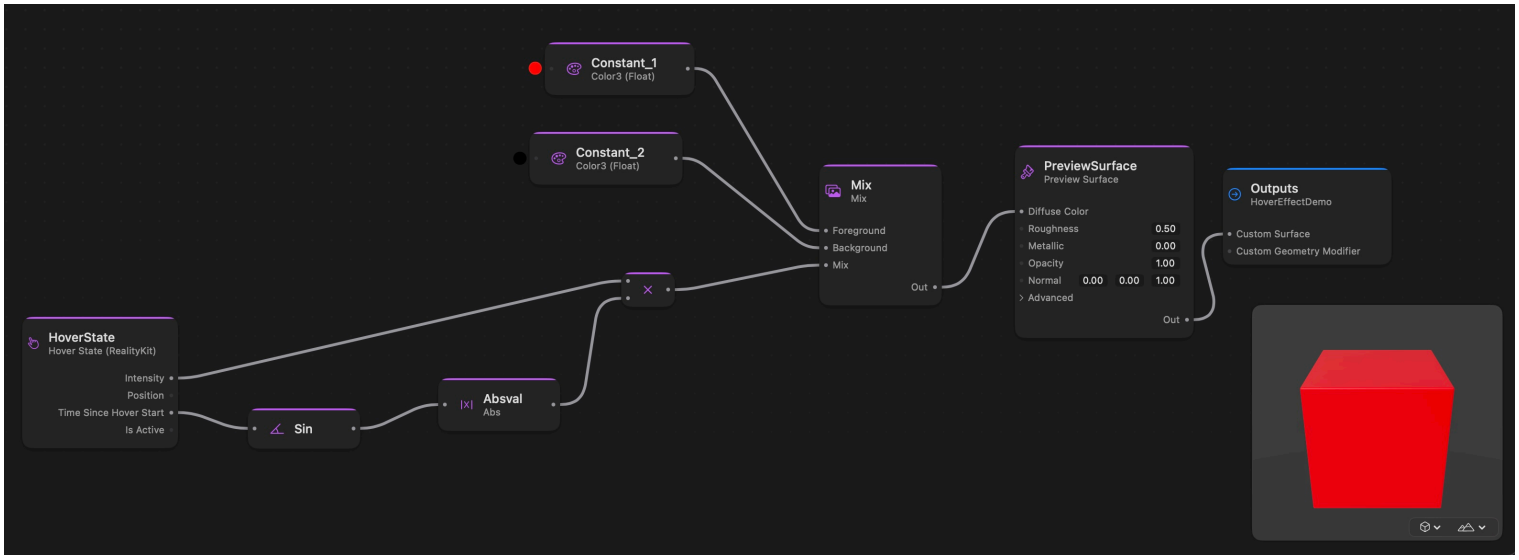
## Shader inputs

Pass `HoverEffectComponent.ShaderHoverEffectInputs` to the `HoverEffectComponent` constructor with `shader(_:)` to activate the `HoverState` Shader Graph node in a `ShaderGraphMaterial`. Apply the `ShaderGraphMaterial` to the entity's `ModelComponent` when using this effect.

```
let boxSize = SIMD3<Float>(0.5, 0.1, 0.05)

let customMaterial = try! await ShaderGraphMaterial(
   named: "/Root/HoverEffectComponentDemo",
   from: Bundle.main.url(forResource: "DemoScene", withExtension: "usda")!
)

let entity = ModelEntity(
    mesh: MeshResource.generateBox(size: boxSize),
    materials: [customMaterial]
)

// ...

let hoverComponent = HoverEffectComponent(.shader(
    HoverEffectComponent.ShaderHoverEffectInputs(
```

```
        fadeInDuration: 1.0, fadeOutDuration: 1.0
    )
))
entity.components.set(hoverComponent)
```

This shader graph represents a red and blue effect that interpolates between each color using a sine wave.




Hovering over the entity displays the following effect.

Play ▶

# Topics

## Structures

`struct GroupID`

A struct for grouping hover effects together.

`struct HighlightHoverEffectStyle`

A type that configures the visual aspects of a highlight hover effect for an entity

`struct HoverEffect`

An effect that applies when a person looks at or directly touches the entity.

`struct ShaderHoverEffectInputs`

A type that configures the visual aspects of a hover effect that applies a custom material

`struct SpotlightHoverEffectStyle`

A type that configures the visual aspects of a spotlight hover effect for an entity.

### Initializers

`init()`

Creates a hover effect component with a default spotlight effect.

`init(HoverEffectComponent.HoverEffect)`

Creates a hover effect component with a particular visualization effect.

### Instance Properties

`var hoverEffect: HoverEffectComponent.HoverEffect`

The current hover effect of the hover effect component.

### Enumerations

`enum OpacityFunction`

The blending technique options a hover effect applies to its entity's base material.

---

## Relationships

### Conforms To

`Component`

---

## See Also

## Visual adjustments

`struct` `BillboardComponent`

A component that orients an entity instance so that it continuously points toward the active camera.

`struct` `EnvironmentBlendingComponent`

A component that controls how an entity blends visually with objects in the local environment.

`struct` `LensDistortionData`

A description of estimated lens distortion that can be used to rectify images.

`struct` `ImagePresentationComponent`

A component that supports general image presentation.