#WWDC19

Introducing RealityKit and Reality Composer

Cody White, Apple Inc.
Tyler Casella, Apple Inc.

Introduction

Incredible variety of AR apps

Developers have many common needs

- Rendering
- Physics
- Animation

Building Apps for AR

Content interacts with the real world

Attach content to physical objects

Virtual content can influence the real world

Match the real environment

Building Apps for AR

Content interacts with the real world

Attach content to physical objects

Virtual content can influence the real world

Match the real environment

RealityKit

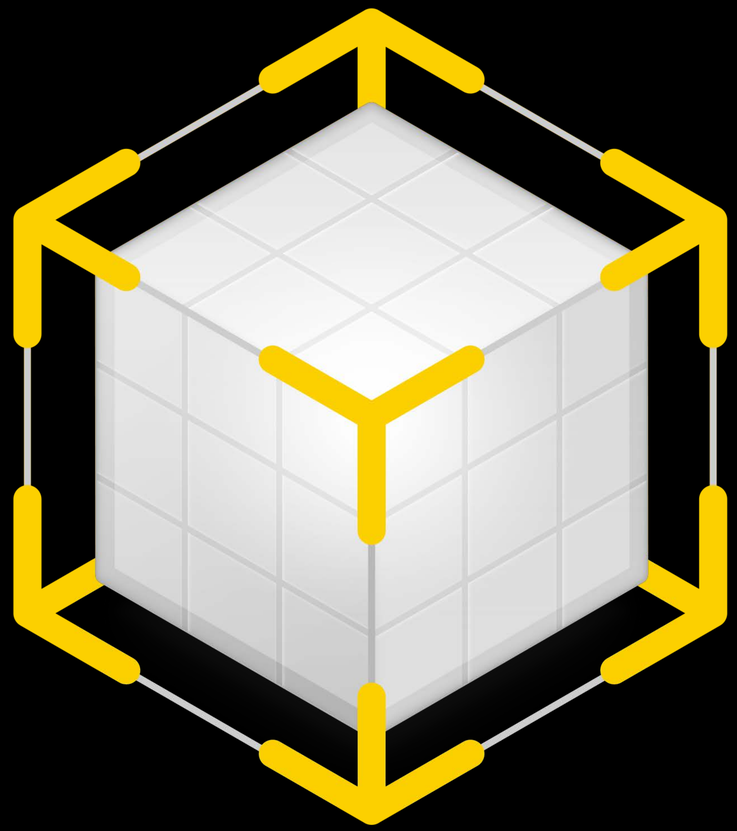
AR First

Realistic rendering and simulation

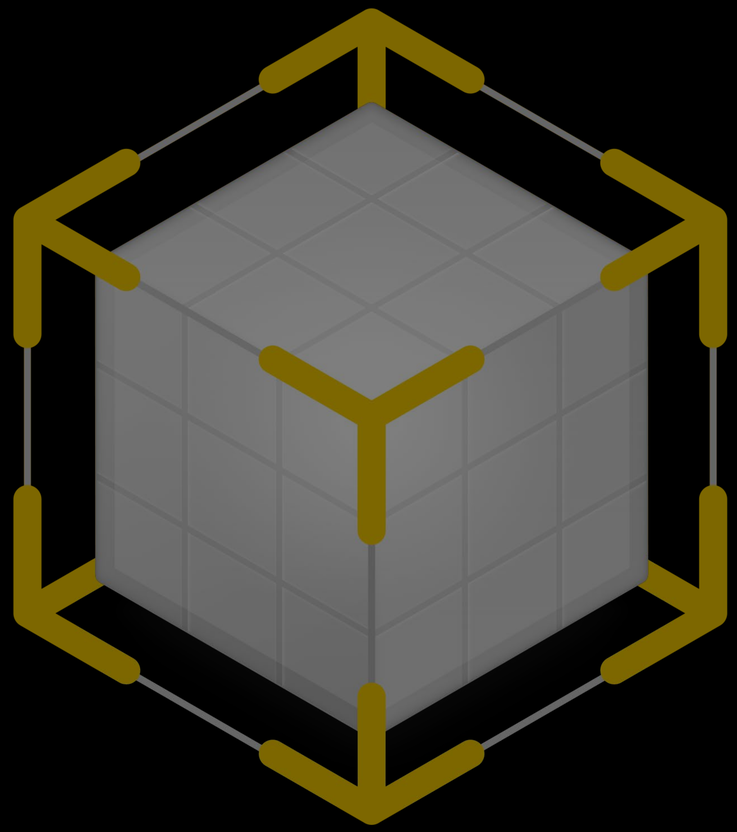
Designed for Swift

iOS and macOS

RealityKit and Reality Composer



RealityKit and Reality Composer



```
import UIKit
import RealityKit

class ViewController: UIViewController {

    @IBOutlet var arView: ARView!

    override func viewDidLoad() {
        super.viewDidLoad()

        let anchor = AnchorEntity(plane: .horizontal)
        arView.scene.addAnchor(anchor)

        let flyer = try Entity.loadModel(named: "flyer")
        anchor.addChild(flyer)

    }
}
```

```
import UIKit
import RealityKit

class ViewController: UIViewController {

    @IBOutlet var arView: ARView!

    override func viewDidLoad() {
        super.viewDidLoad()

        let anchor = AnchorEntity(plane: .horizontal)
        arView.scene.addAnchor(anchor)

        let flyer = try Entity.loadModel(named: "flyer")
        anchor.addChild(flyer)

    }
}
```

Systems and Framework Basics

Rendering

Animation

Physics

Synchronization

ECS

Audio

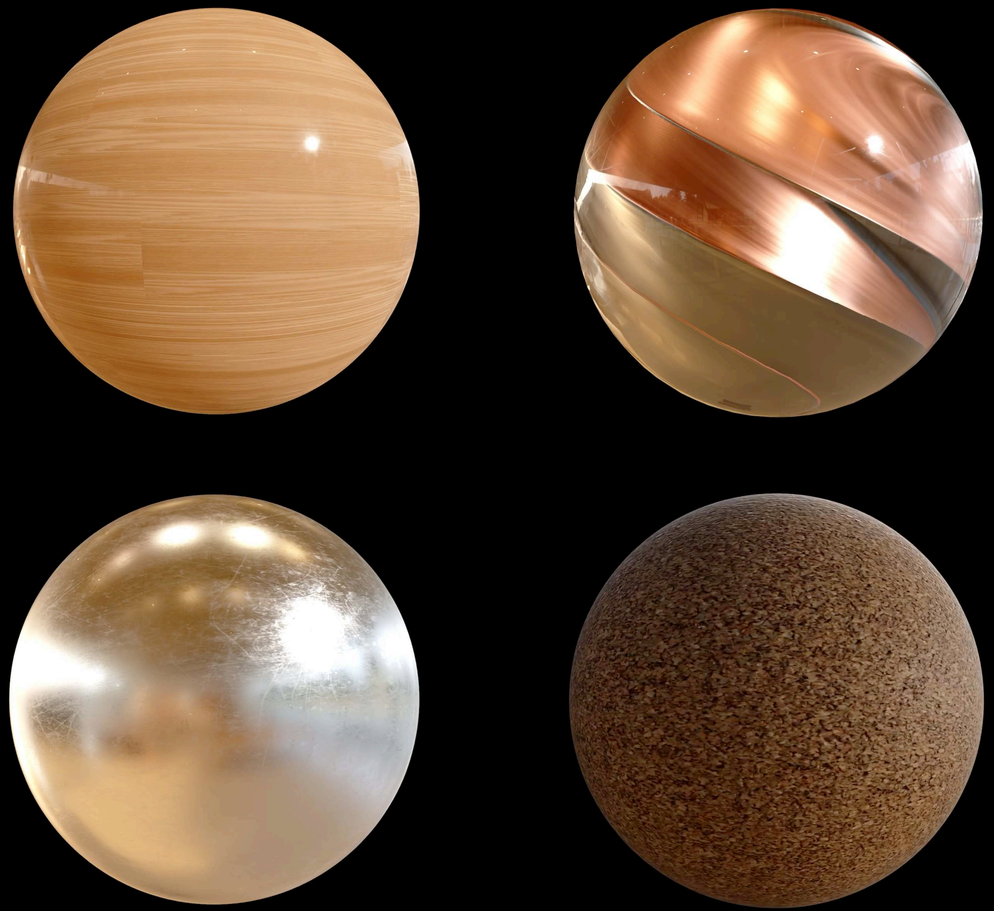
Rendering

Physically-based shading

Built with Metal

- Optimized for Apple GPUs
- Latest features

AR-focused



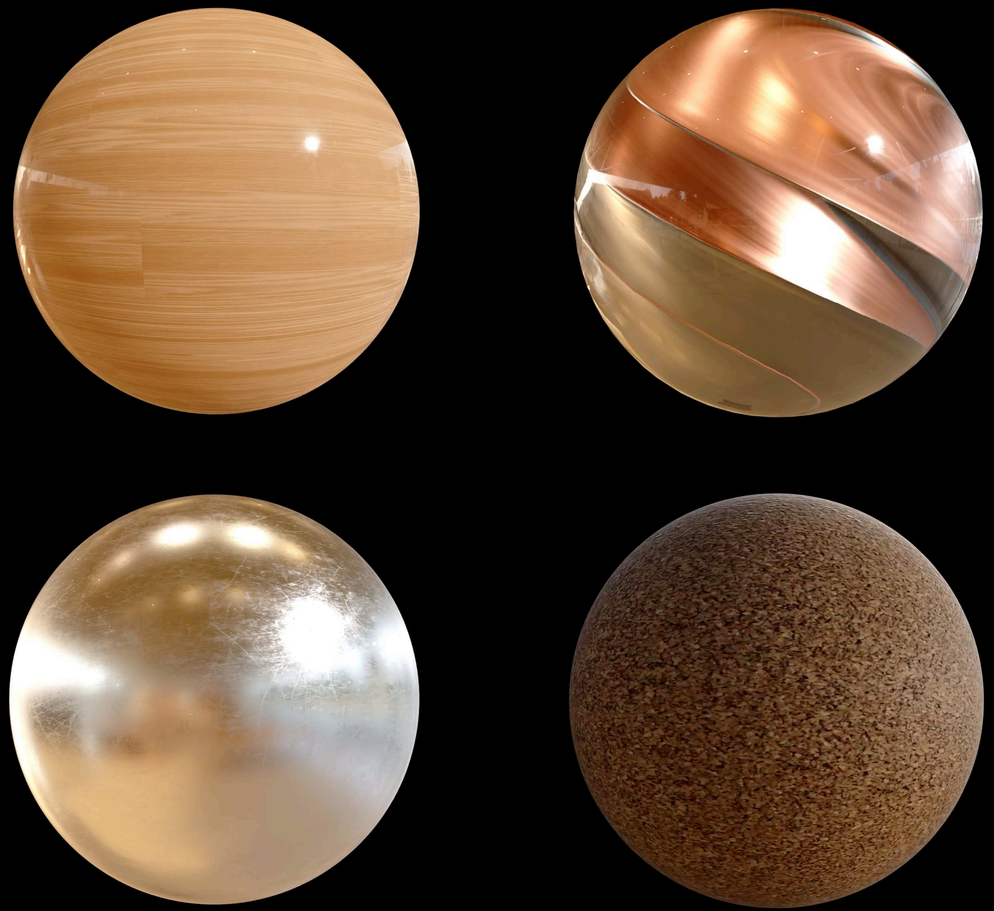
Rendering

Physically-based shading

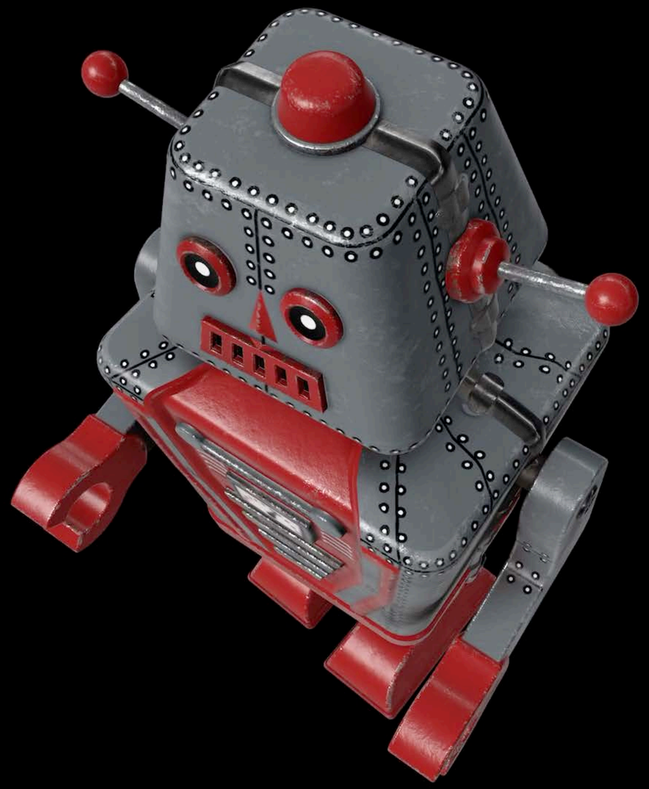
Built with Metal

- Optimized for Apple GPUs
- Latest features

AR-focused



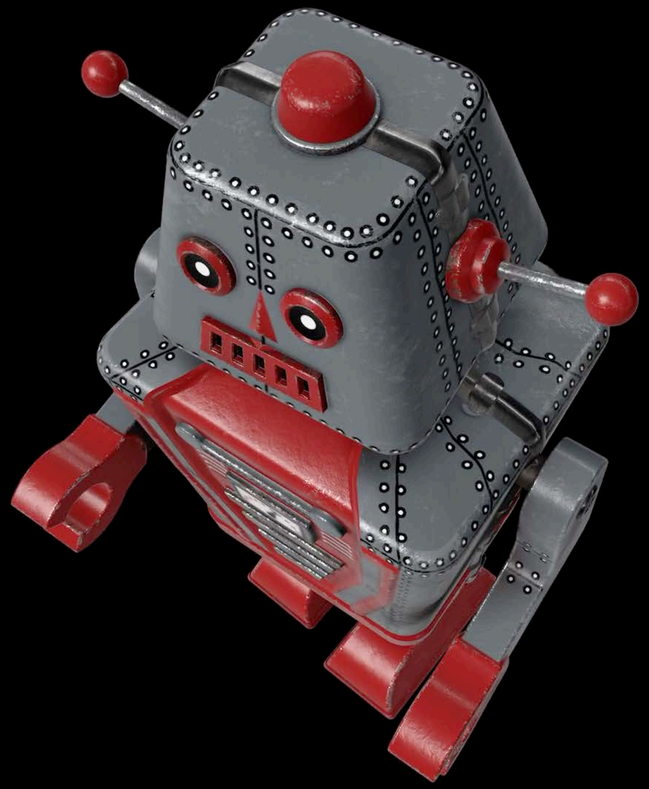
Animation



Skeletal

Transform

Animation



Skeletal

Transform

Physics

Collision detection Rigid body dynamics AR-enabled

Physics

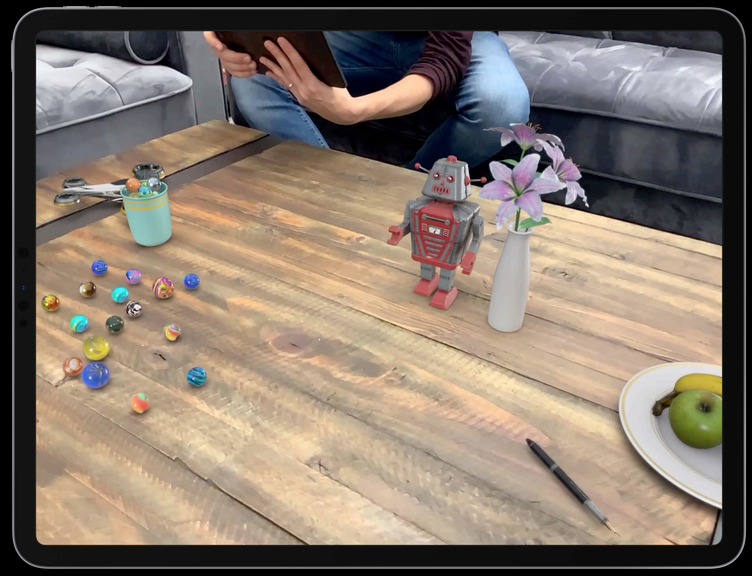
Collision detection Rigid body dynamics AR-enabled

Synchronization

Multi-device

Collaborative map building with ARKit

Multiplayer

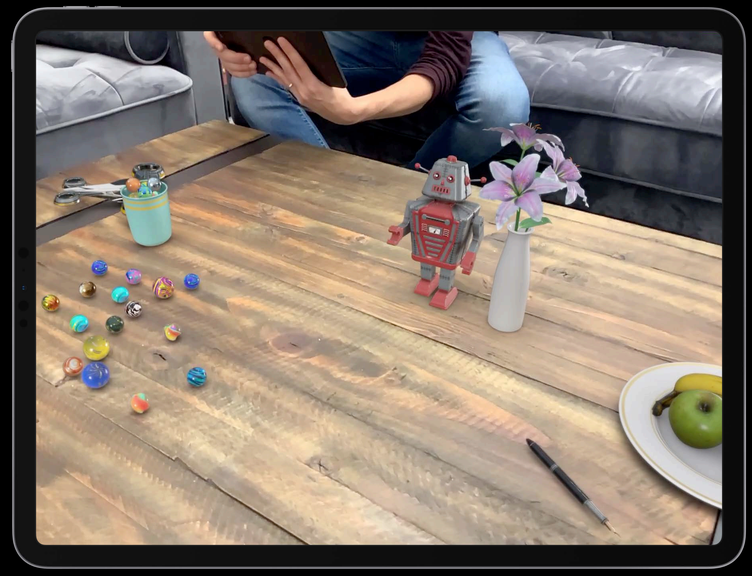


Synchronization

Multi-device

Collaborative map building with ARKit

Multiplayer



Entity-Component System

Entities and components

Composition versus inheritance

Extensible

Automatic network sync

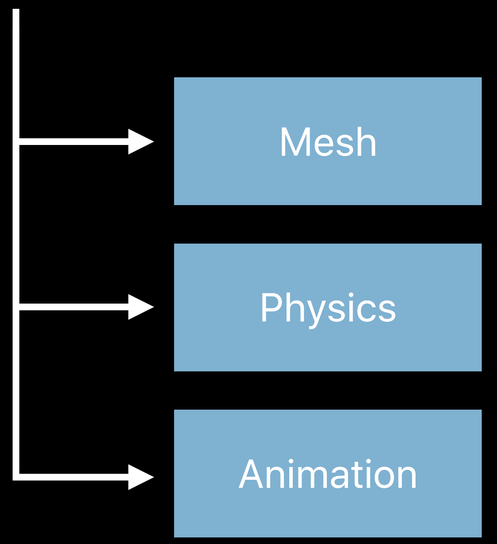
Entity-Component System

Entities and components

Composition versus inheritance

Extensible

Automatic network sync



Audio

Spatial understanding

Automatic listener configuration

Audio playback on tracked 3D objects

Audio

Spatial understanding

Automatic listener configuration

Audio playback on tracked 3D objects

Reality File

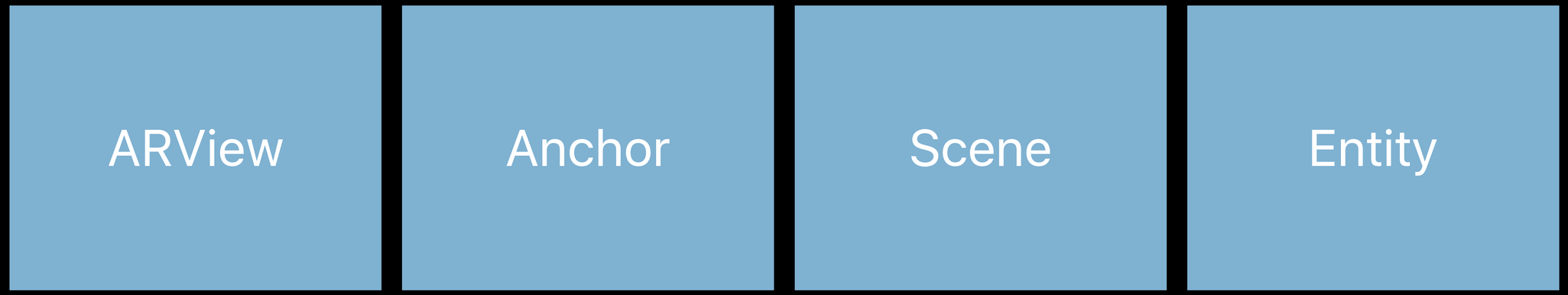
Contains all data needed for rendering and simulation

Optimized for RealityKit

Exported from Reality Composer

Preview in AR Quick Look

Framework Basics



ARView

Sets up the environment

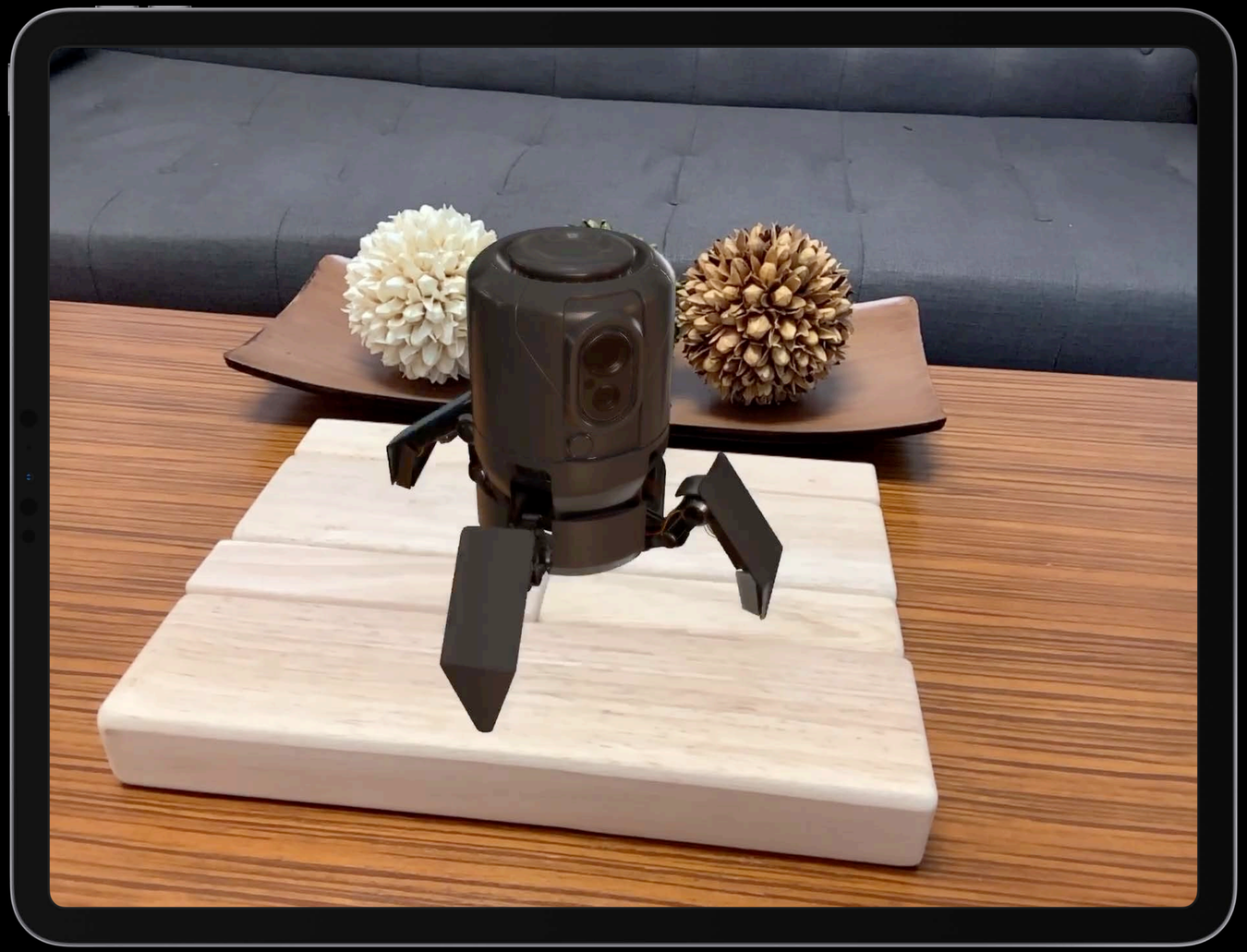
Handles gestures

Focus on the app

Realistic camera effects

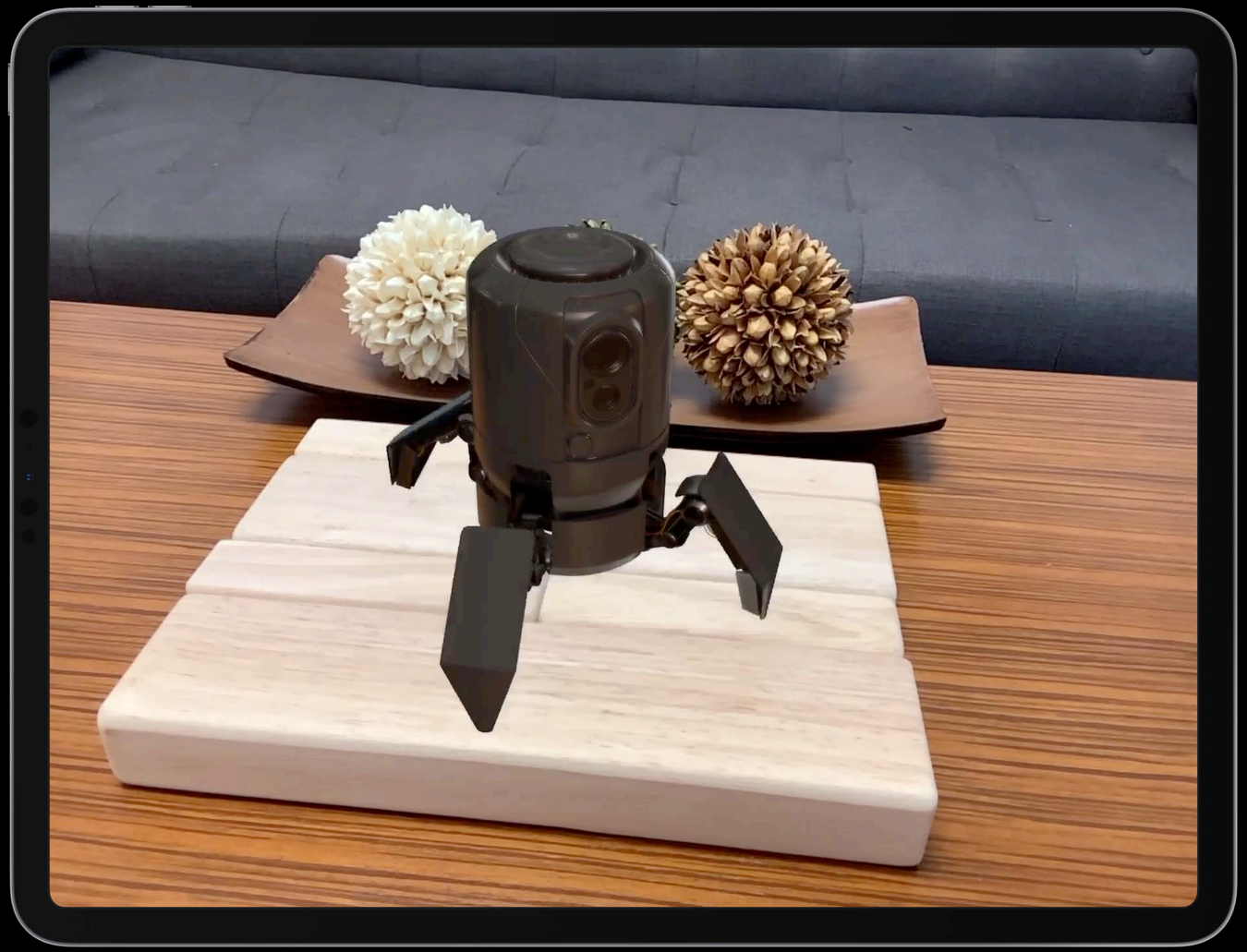
Shadowing

ARView



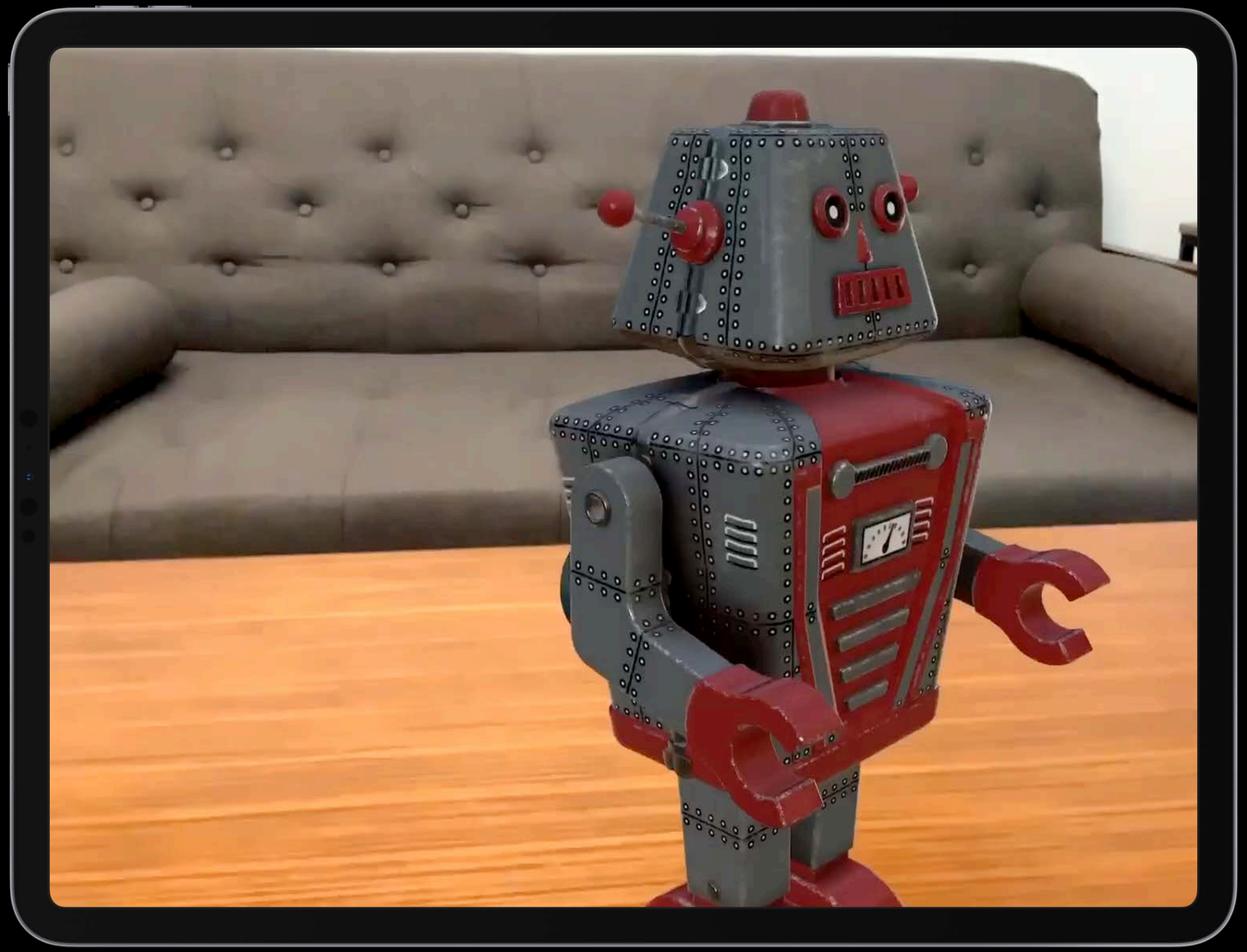
Shadowing

ARView



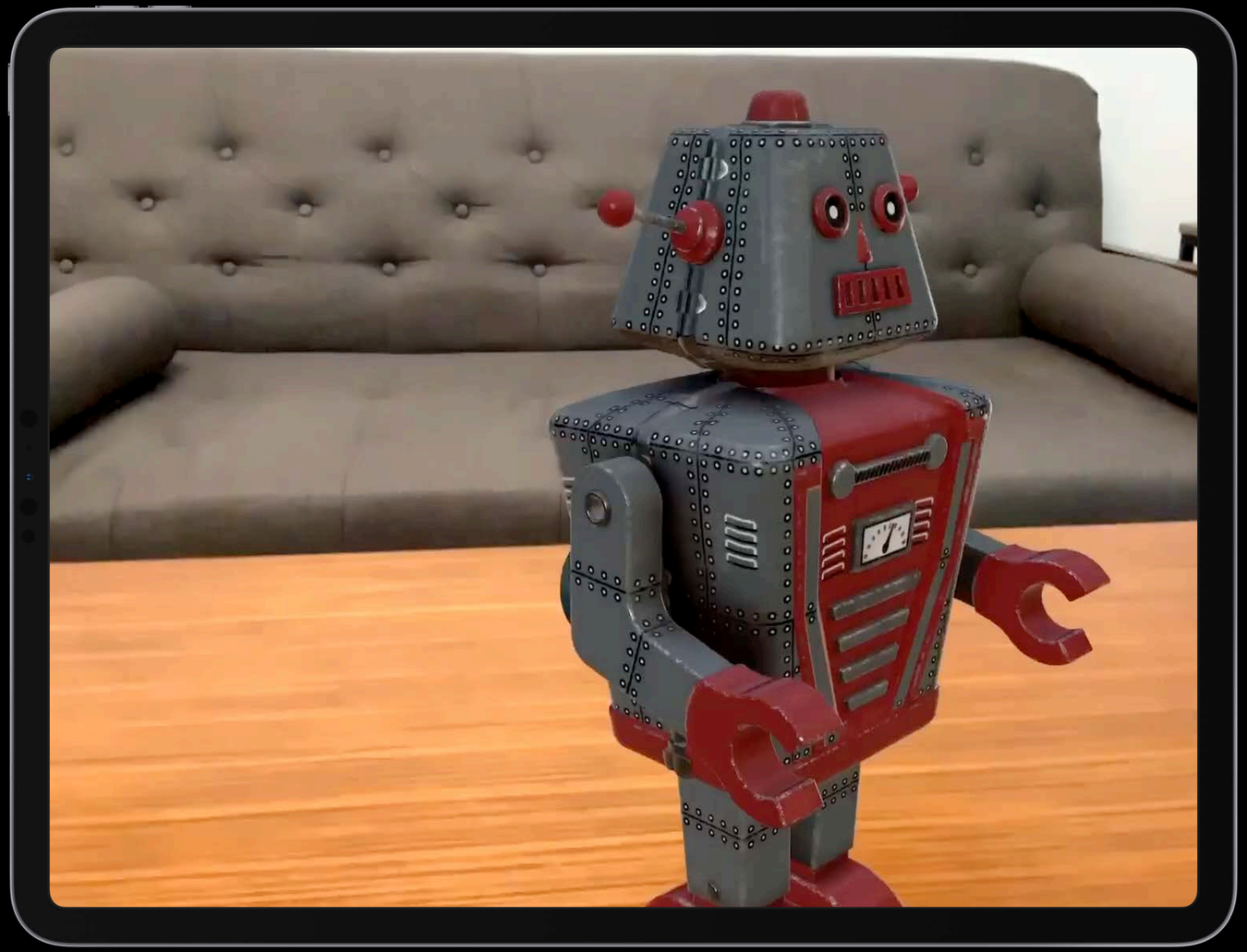
Motion Blur

ARView



Motion Blur

ARView



Depth of Field

ARView

Depth of Field

ARView

Camera Noise

ARView

Camera Noise

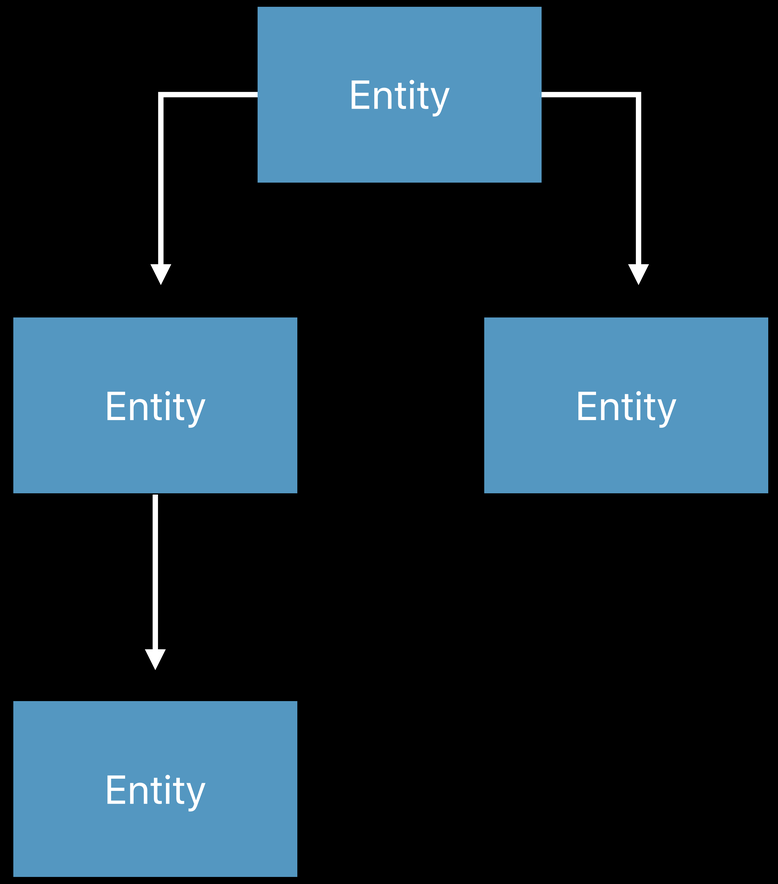
ARView

Entity

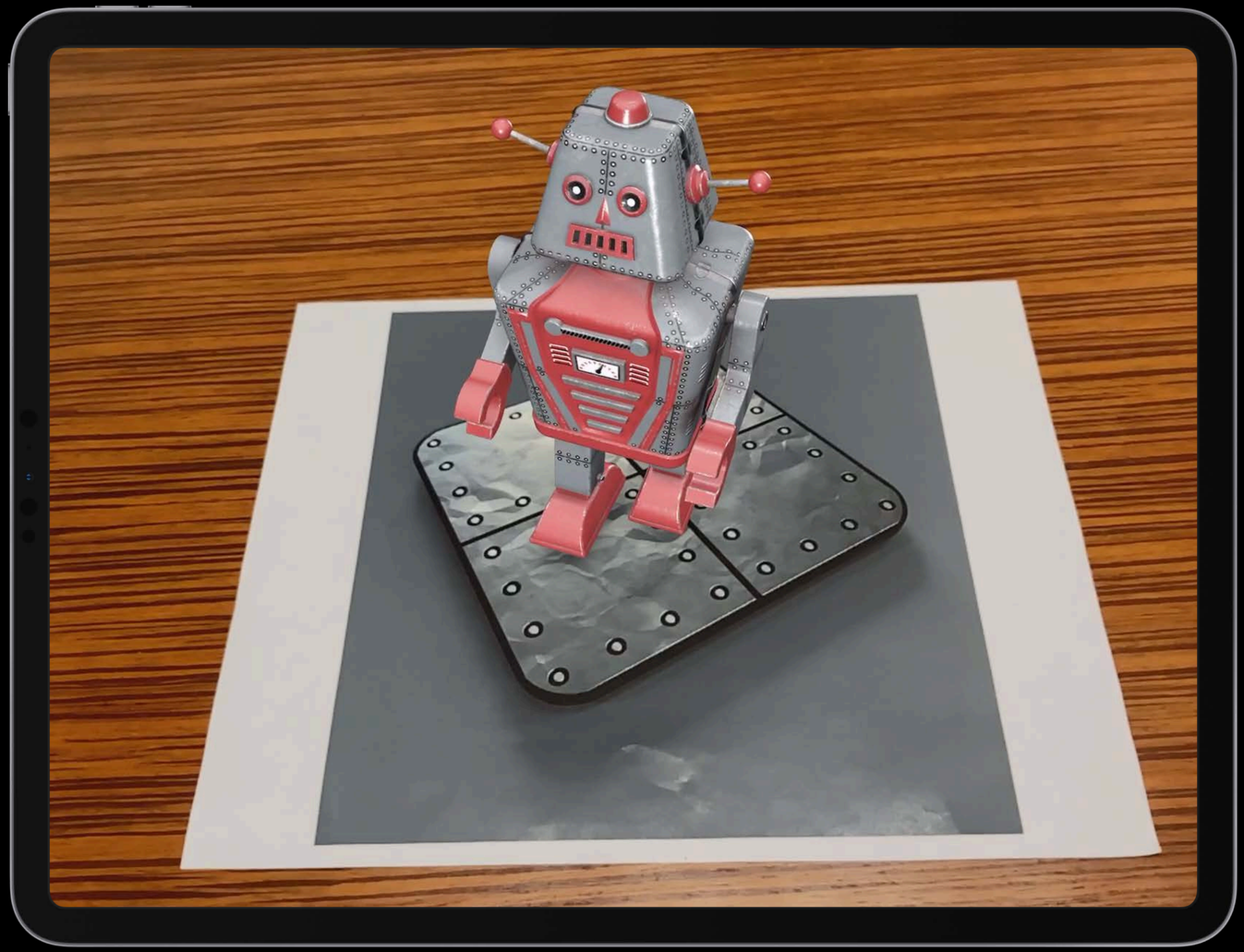
Building block of every AR object

Establishes scene structure

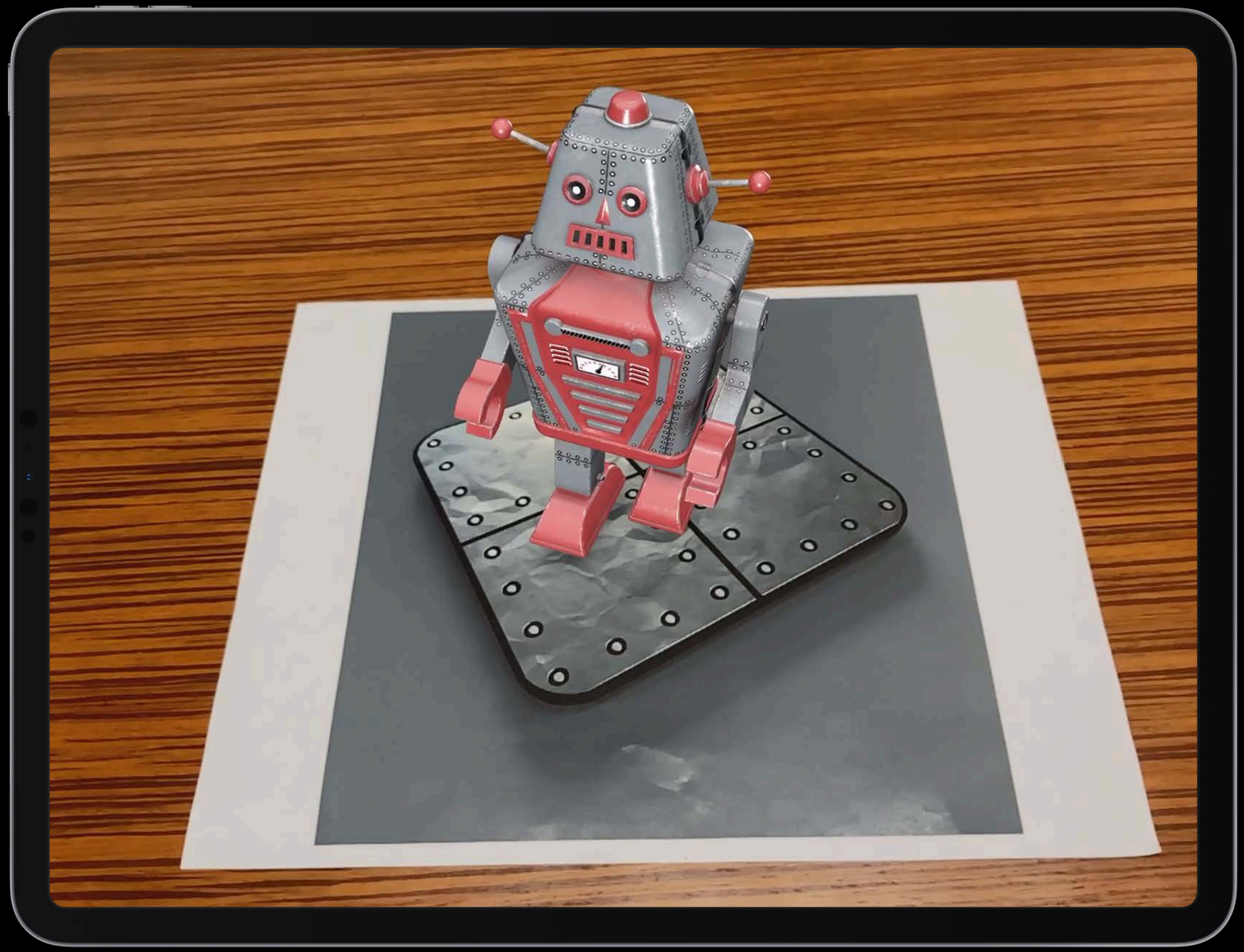
Provides transform hierarchy



AR Anchoring

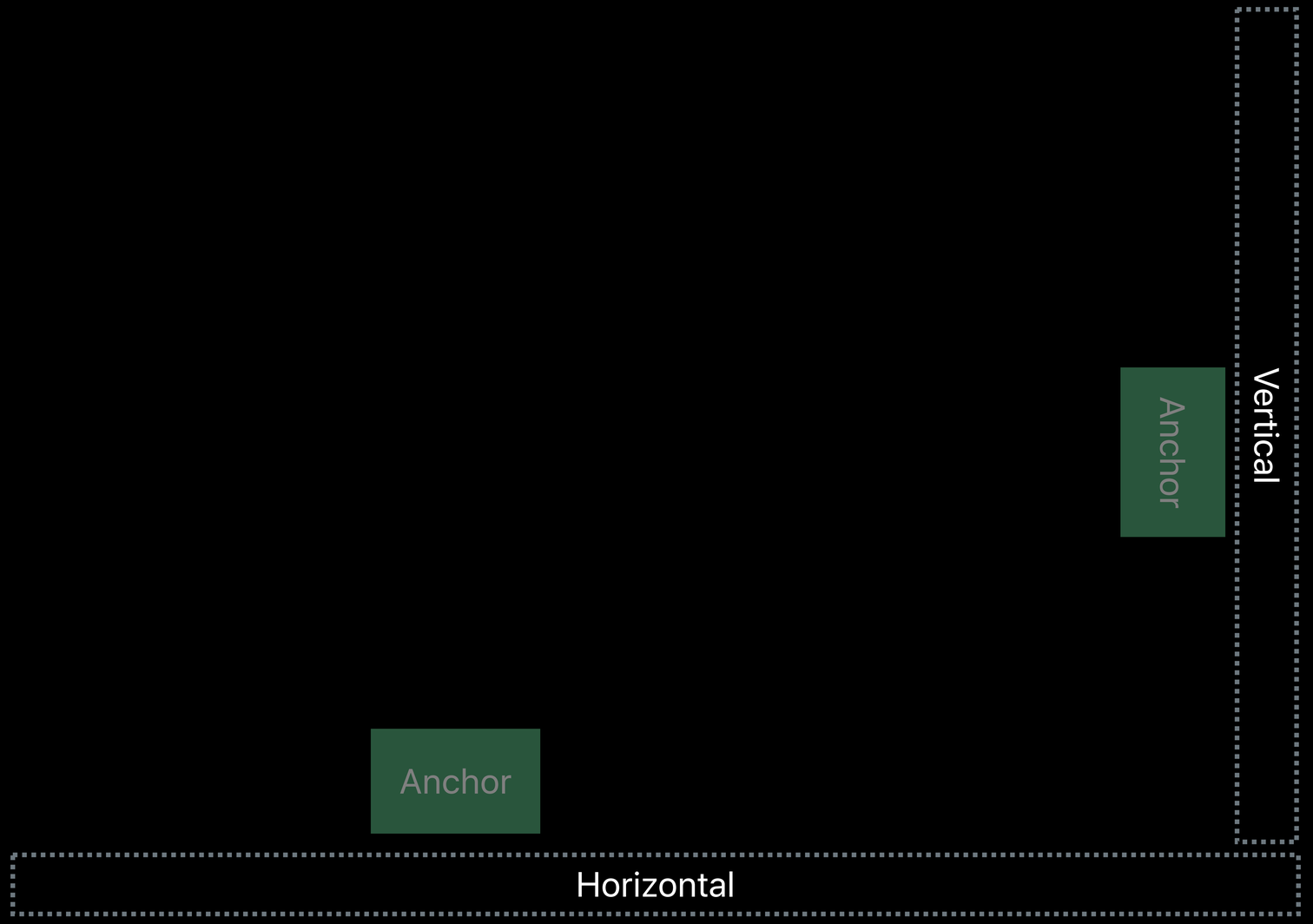


AR Anchoring

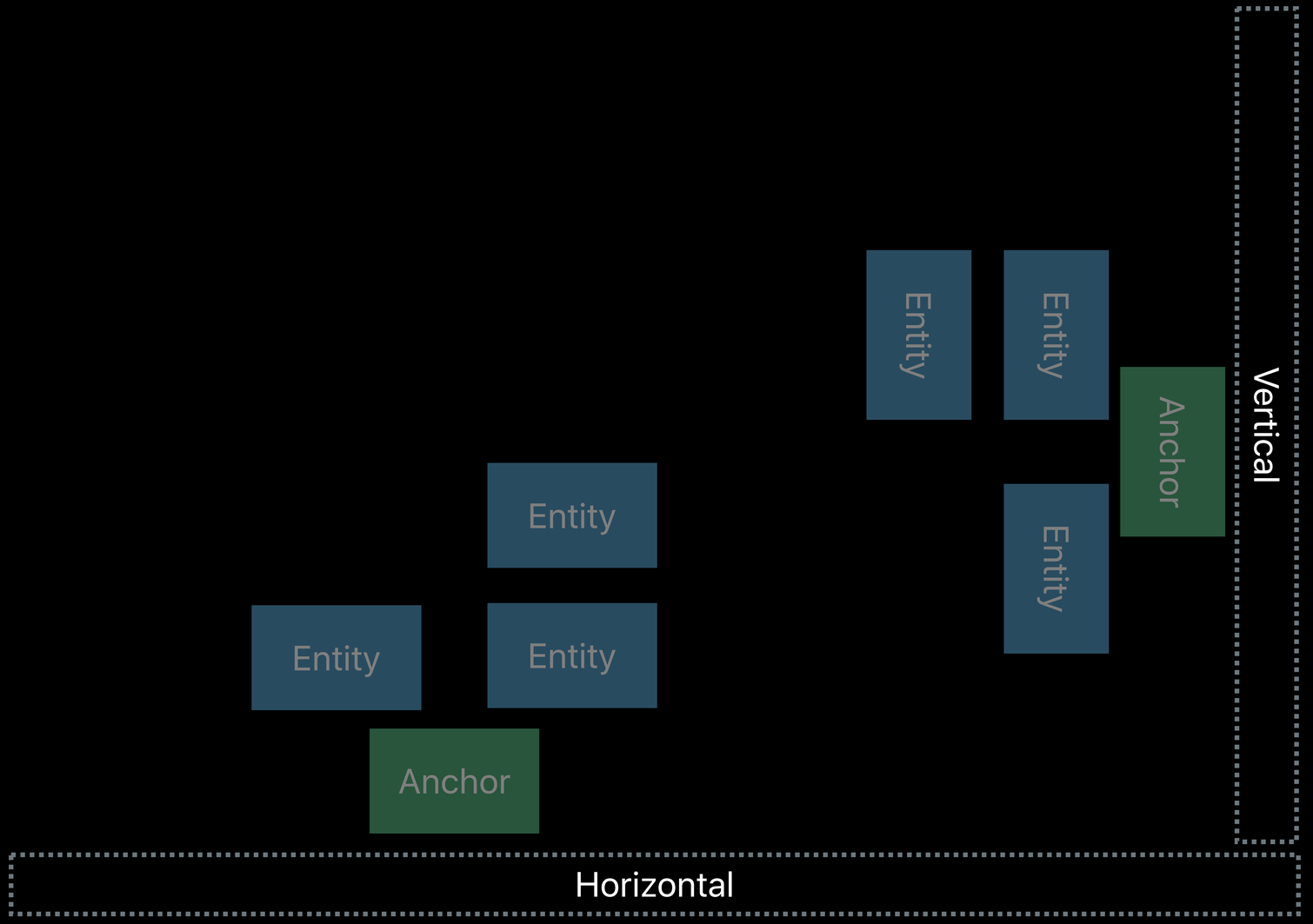


AR Anchoring

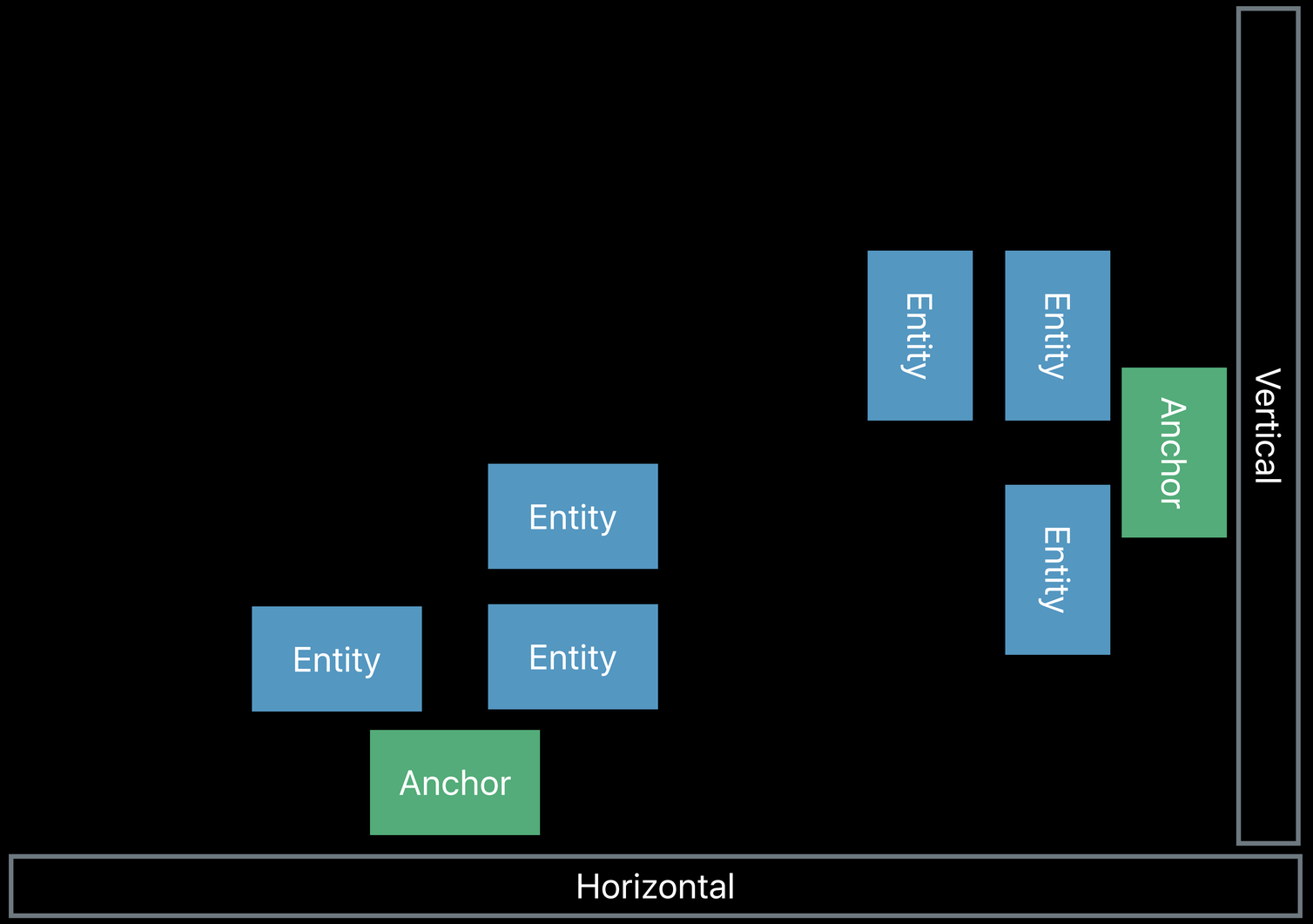
AR Anchoring



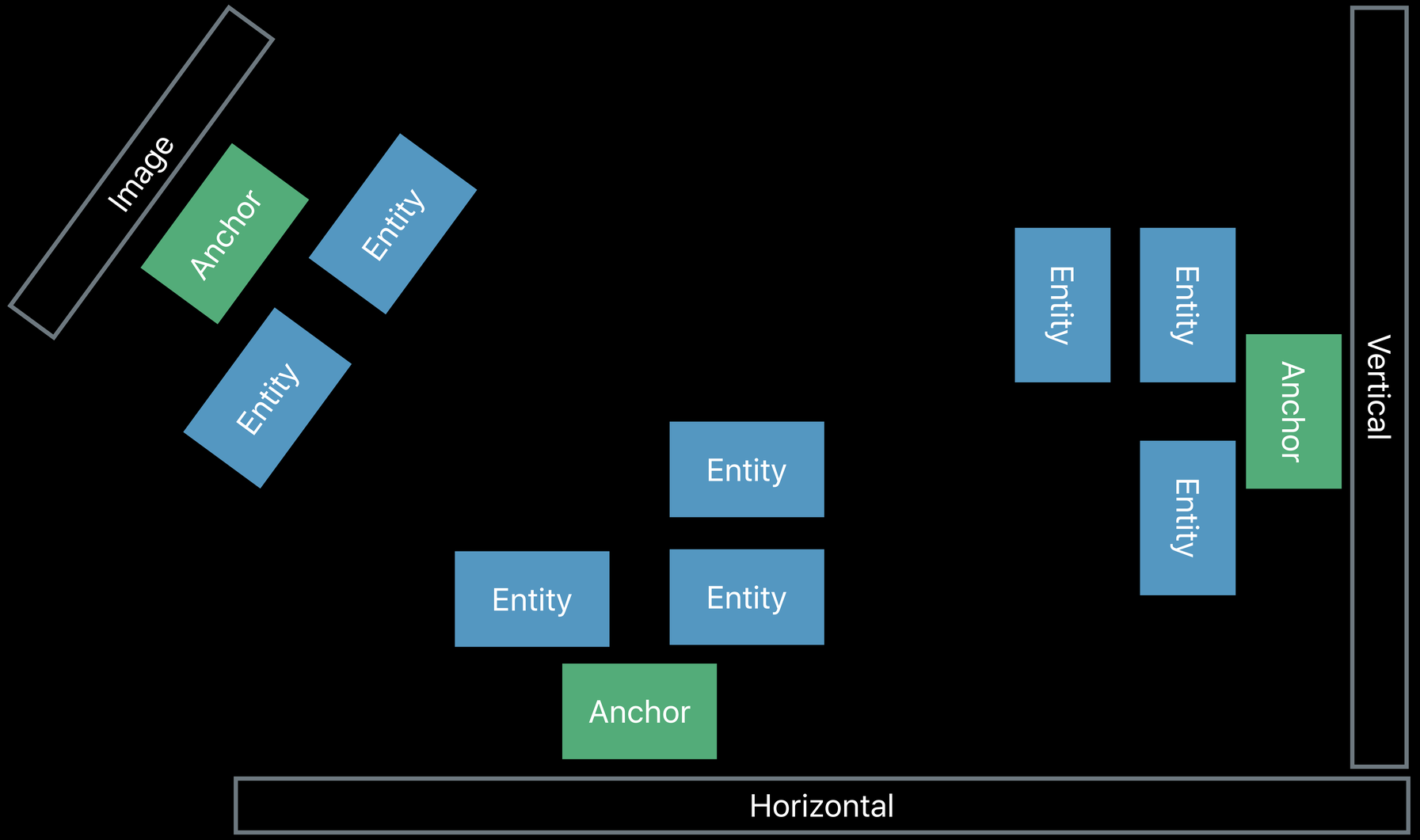
AR Anchoring



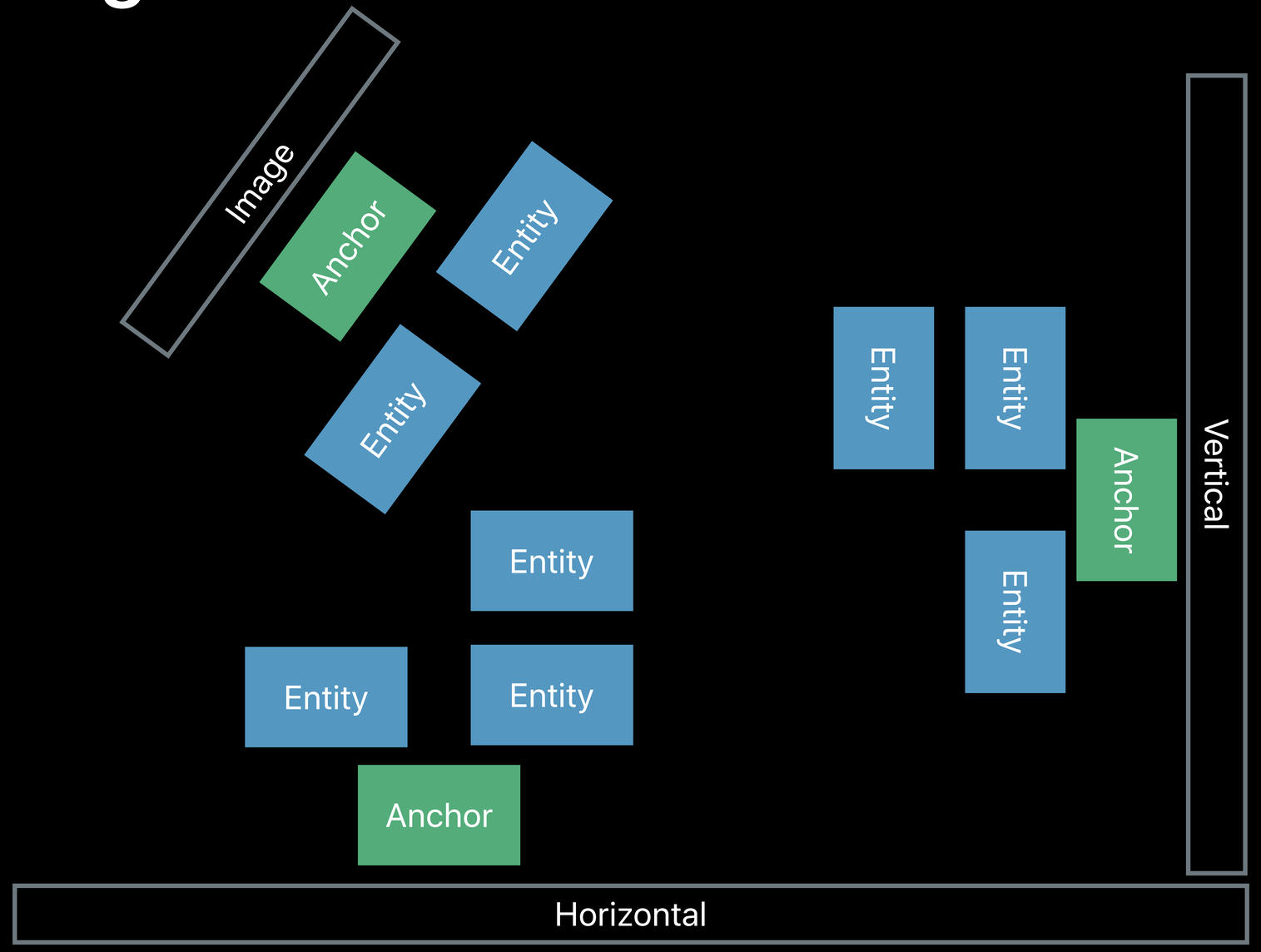
AR Anchoring



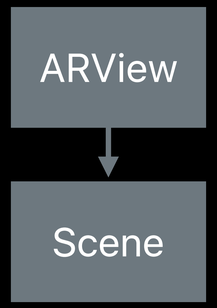
AR Anchoring

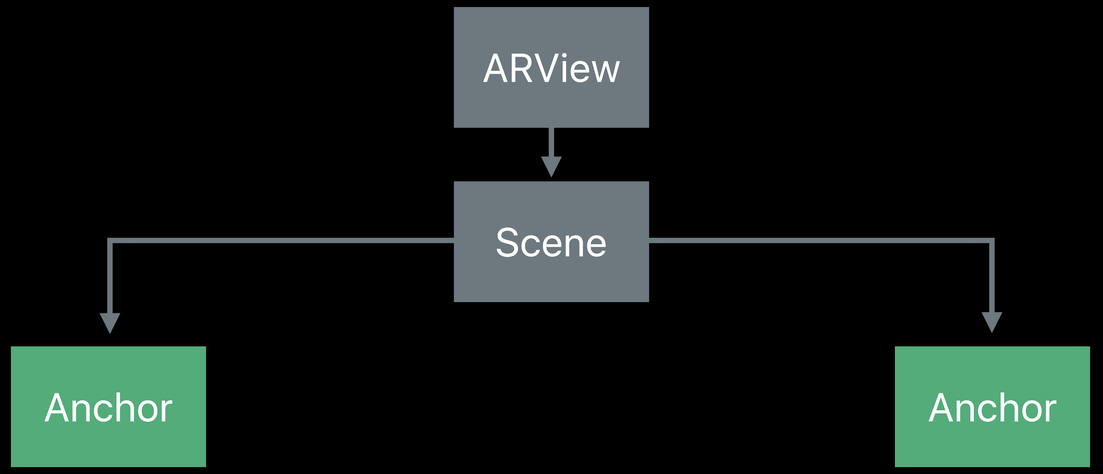


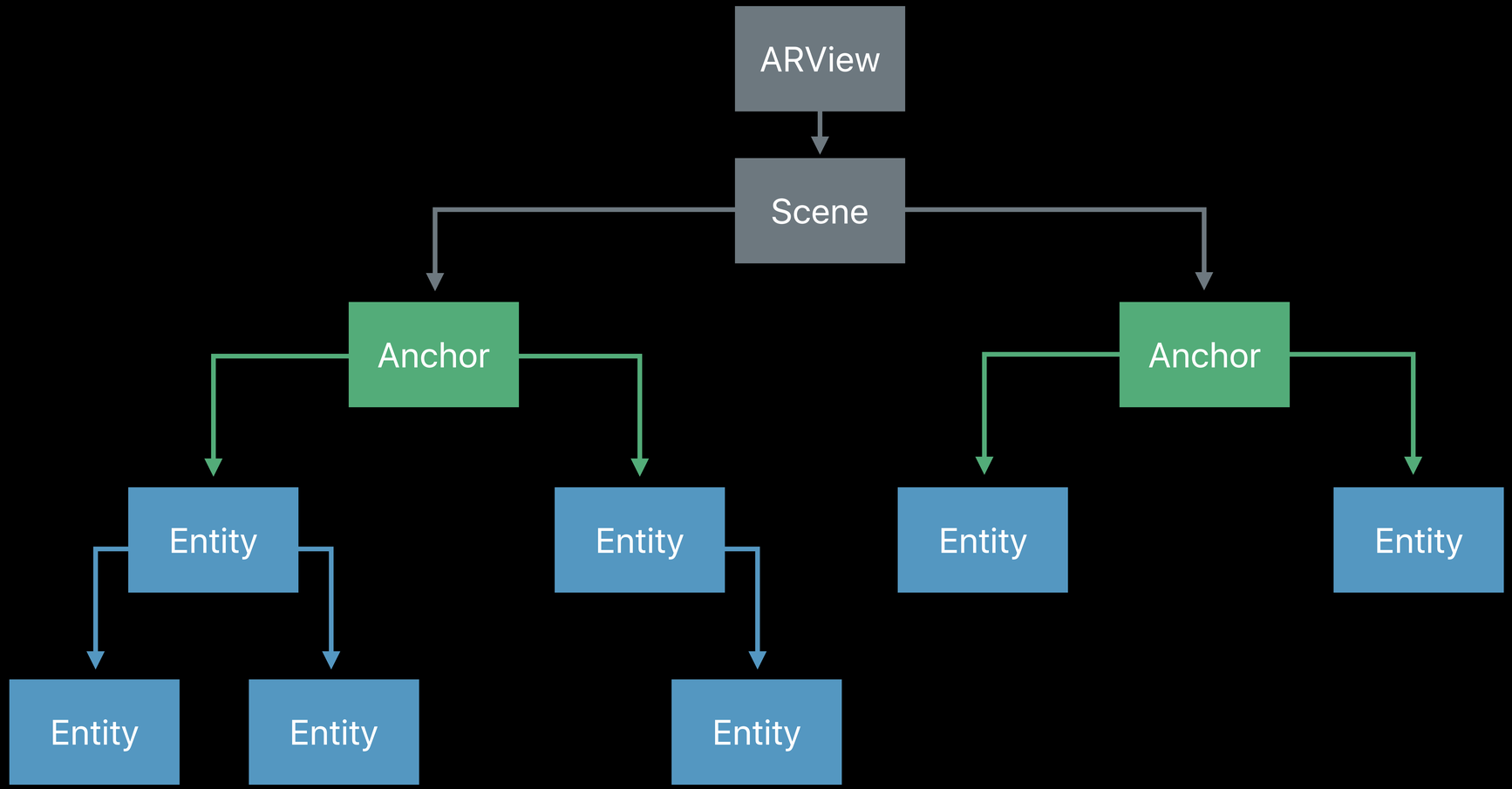
AR Anchoring



ARView


ARView
Scene


ARView
Scene
Anchor
Anchor

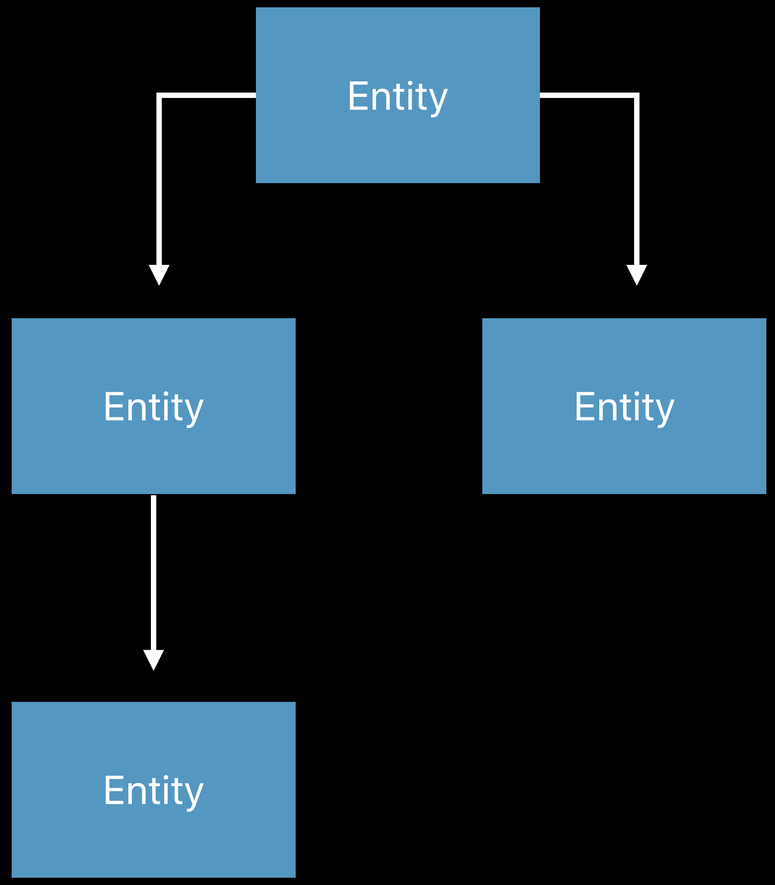

ARView
Scene
Anchor
Anchor
Entity
Entity
Entity
Entity
Entity
Entity
Entity

Demo

Diving Deeper with RealityKit

Tyler Casella, Apple Inc.

Entities and Components

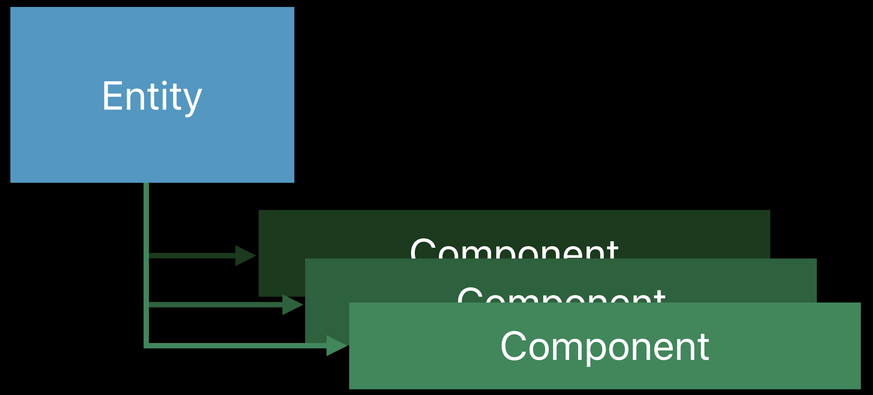


Entities and Components

Composition over inheritance

Promotes reuse

Flexible and scalable



Entities and Components

Entities and Components

Ball

Lamp

Camera

Entities and Components

Anchoring

Ball

Anchoring

Lamp

Anchoring

Camera

Entities and Components

Anchoring

Ball

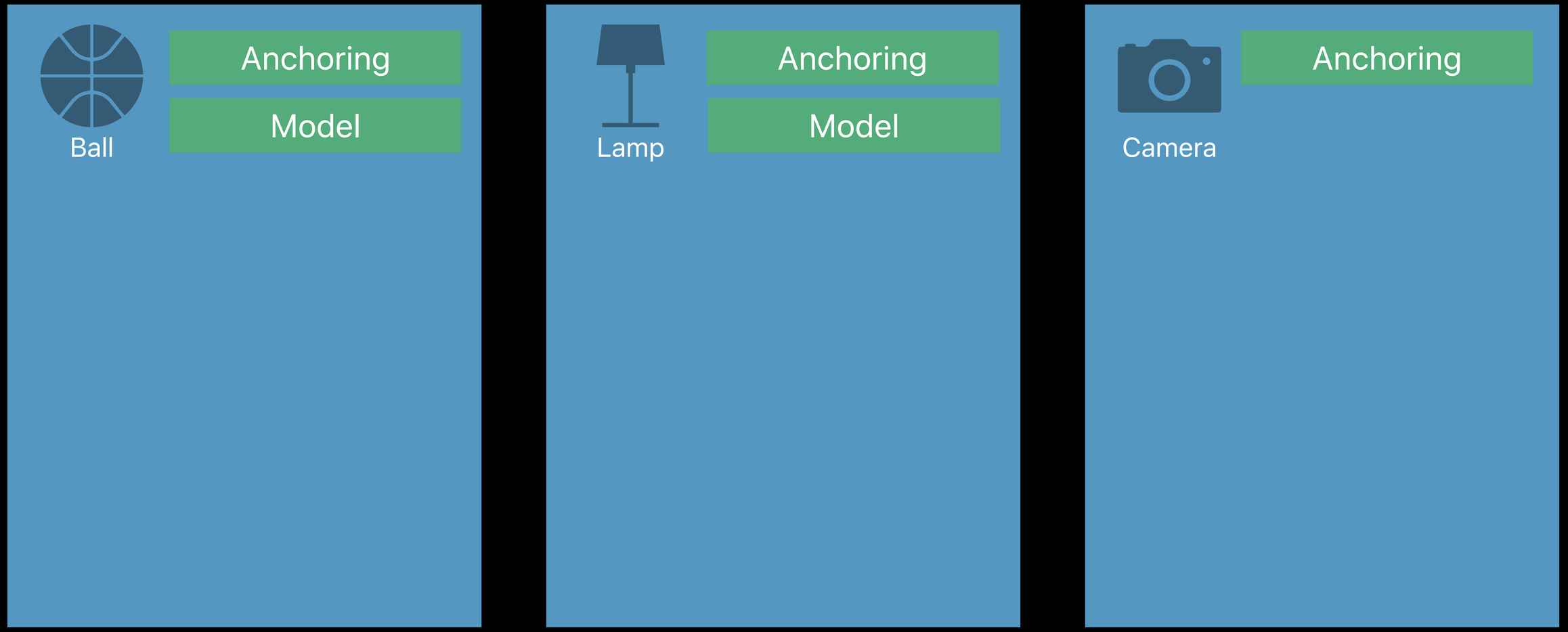
Anchoring

Lamp

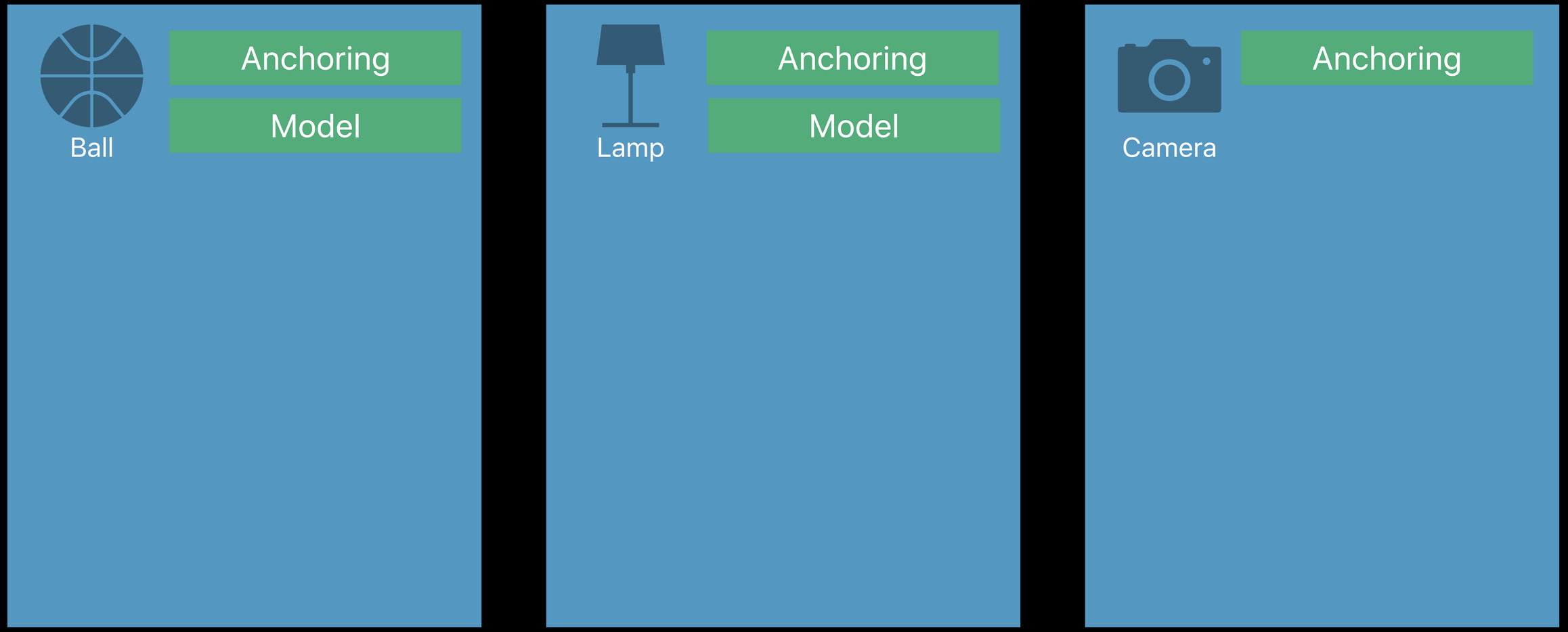
Anchoring

Camera

Entities and Components



Entities and Components



Entities and Components

Ball

Anchoring

Model

Collision

Lamp

Anchoring

Model

Collision

Camera

Anchoring

Entities and Components

Ball

Anchoring

Model

Collision

Lamp

Anchoring

Model

Collision

Camera

Anchoring

Entities and Components

Ball

- Anchoring
- Model
- Collision
- Physics

Lamp

- Anchoring
- Model
- Collision
- SpotLight
- Shadow

Camera

- Anchoring
- PerspectiveCamera

```swift
/// Create new entity
let entity = Entity( )

/// Add/modify component
entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)
entity.components[SpotLightComponent.Shadow.self] = SpotLightComponent.Shadow()

/// Remove component
entity.components[CollisionComponent.self] = nil

/// Add child entity
entity.addChild(childEntity)

/// Set local position
entity.position = [1.0, 0.0, 0.5]

/// Set world position
entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
```

```
/// Create new entity
let entity = Entity( )

/// Add/modify component
entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)
entity.components[SpotLightComponent.Shadow.self] = SpotLightComponent.Shadow()

/// Remove component
entity.components[CollisionComponent.self] = nil

/// Add child entity
entity.addChild(childEntity)

/// Set local position
entity.position = [1.0, 0.0, 0.5]

/// Set world position
entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
```

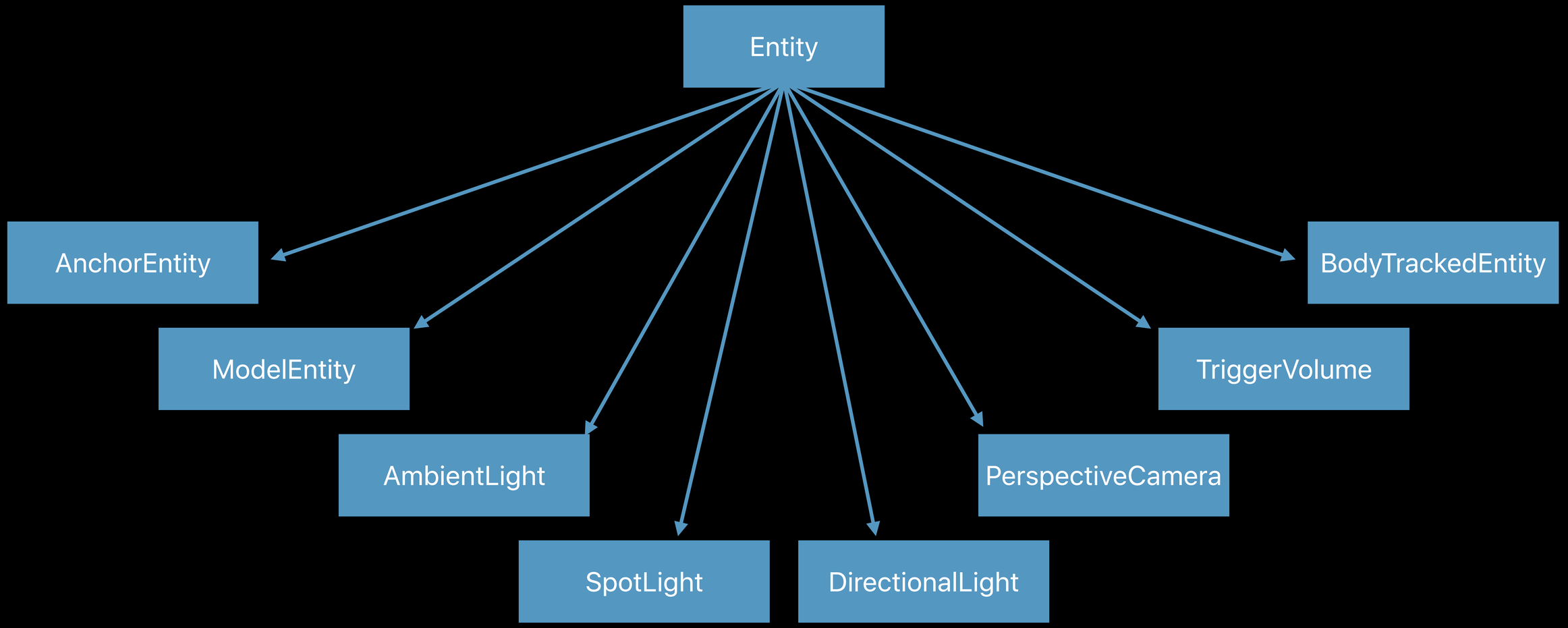
```
/// Create new entity
let entity = Entity( )

/// Add/modify component
entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)
entity.components[SpotLightComponent.Shadow.self] = SpotLightComponent.Shadow()

/// Remove component
entity.components[CollisionComponent.self] = nil

/// Add child entity
entity.addChild(childEntity)

/// Set local position
entity.position = [1.0, 0.0, 0.5]

/// Set world position
entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
```

```
/// Create new entity
let entity = Entity( )

/// Add/modify component
entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)
entity.components[SpotLightComponent.Shadow.self] = SpotLightComponent.Shadow()

/// Remove component
entity.components[CollisionComponent.self] = nil

/// Add child entity
entity.addChild(childEntity)

/// Set local position
entity.position = [1.0, 0.0, 0.5]

/// Set world position
entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
```

```
/// Create new entity
let entity = Entity( )

/// Add/modify component
entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)
entity.components[SpotLightComponent.Shadow.self] = SpotLightComponent.Shadow()

/// Remove component
entity.components[CollisionComponent.self] = nil

/// Add child entity
entity.addChild(childEntity)

/// Set local position
entity.position = [1.0, 0.0, 0.5]

/// Set world position
entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
```

```swift
/// Create new entity
let entity = Entity( )

/// Add/modify component
entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)
entity.components[SpotLightComponent.Shadow.self] = SpotLightComponent.Shadow()

/// Remove component
entity.components[CollisionComponent.self] = nil

/// Add child entity
entity.addChild(childEntity)

/// Set local position
entity.position = [1.0, 0.0, 0.5]

/// Set world position
entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
```


Entity


Entity
AnchorEntity
ModelEntity
AmbientLight
SpotLight
DirectionalLight
PerspectiveCamera
TriggerVolume
BodyTrackedEntity

AnchorEntity

Attaches to real world objects

Automatically tracks target

Integrated with ARKit

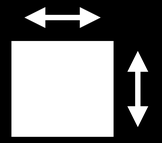
AnchorEntity

Attaches to real world objects

Automatically tracks target

Integrated with ARKit

Anchoring Types



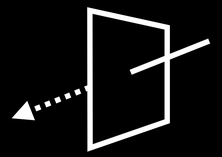
Plane

Face

Camera

Image

Body



ARRaycastResult

Object

World

ARAnchor

```
/// Create an anchor to a table with minimum size
let tableAnchor = AnchorEntity(plane: .horizontal,
                               classification: .table,
                               minimumBounds: [0.5, 0.5])

/// Add table anchor to scene
scene.addAnchor(tableAnchor)


/// Create an anchor to a reference image
let imageAnchor = AnchorEntity(.image(group: "Posters", name: "WorldsFair"))


/// Add image anchor to scene
scene.addAnchor(imageAnchor)
```

```
/// Create an anchor to a table with minimum size
let tableAnchor = AnchorEntity(plane: .horizontal,
                               classification: .table,
                               minimumBounds: [0.5, 0.5])

/// Add table anchor to scene
scene.addAnchor(tableAnchor)

/// Create an anchor to a reference image
let imageAnchor = AnchorEntity(.image(group: "Posters", name: "WorldsFair"))

/// Add image anchor to scene
scene.addAnchor(imageAnchor)
```

```
/// Create an anchor to a table with minimum size
let tableAnchor = AnchorEntity(plane: .horizontal,
                               classification: .table,
                               minimumBounds: [0.5, 0.5])

/// Add table anchor to scene
scene.addAnchor(tableAnchor)
```

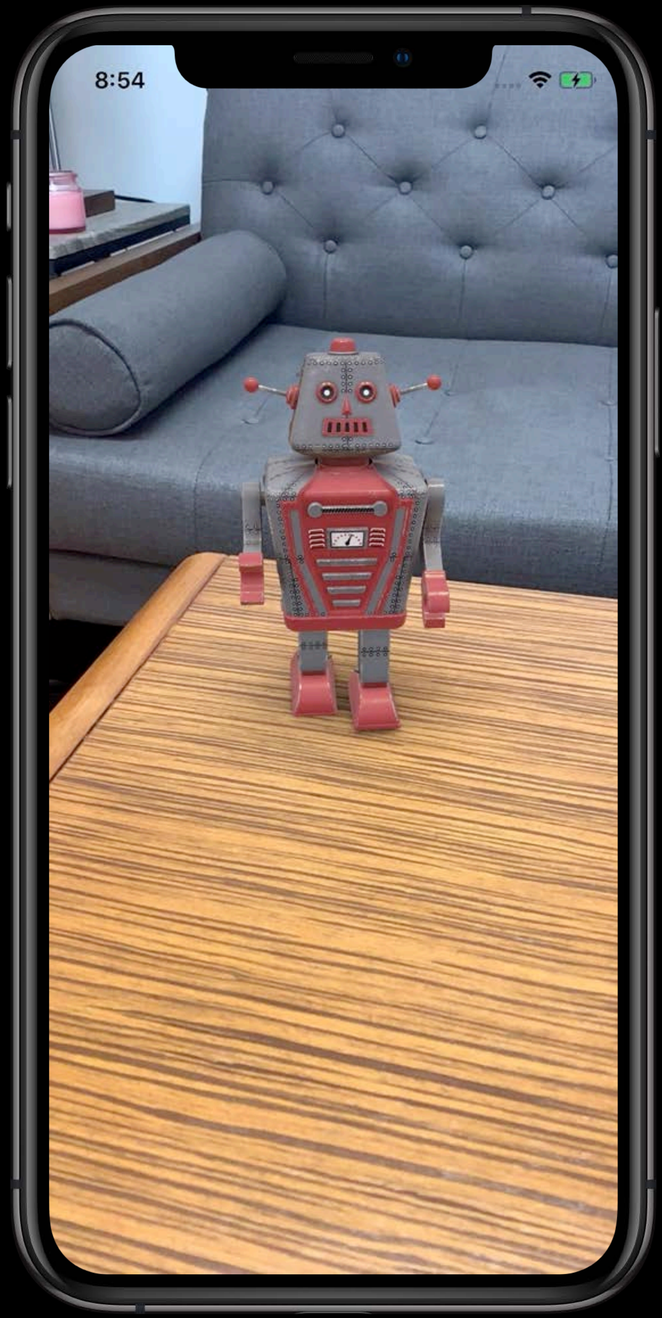
```
/// Create an anchor to a reference image
let imageAnchor = AnchorEntity(.image(group: "Posters", name: "WorldsFair"))


/// Add image anchor to scene
scene.addAnchor(imageAnchor)
```

ModelEntity

Represents visual parts of a scene

Load directly from ‘usdz’ or Reality Files

Contains geometry, animation, and physics

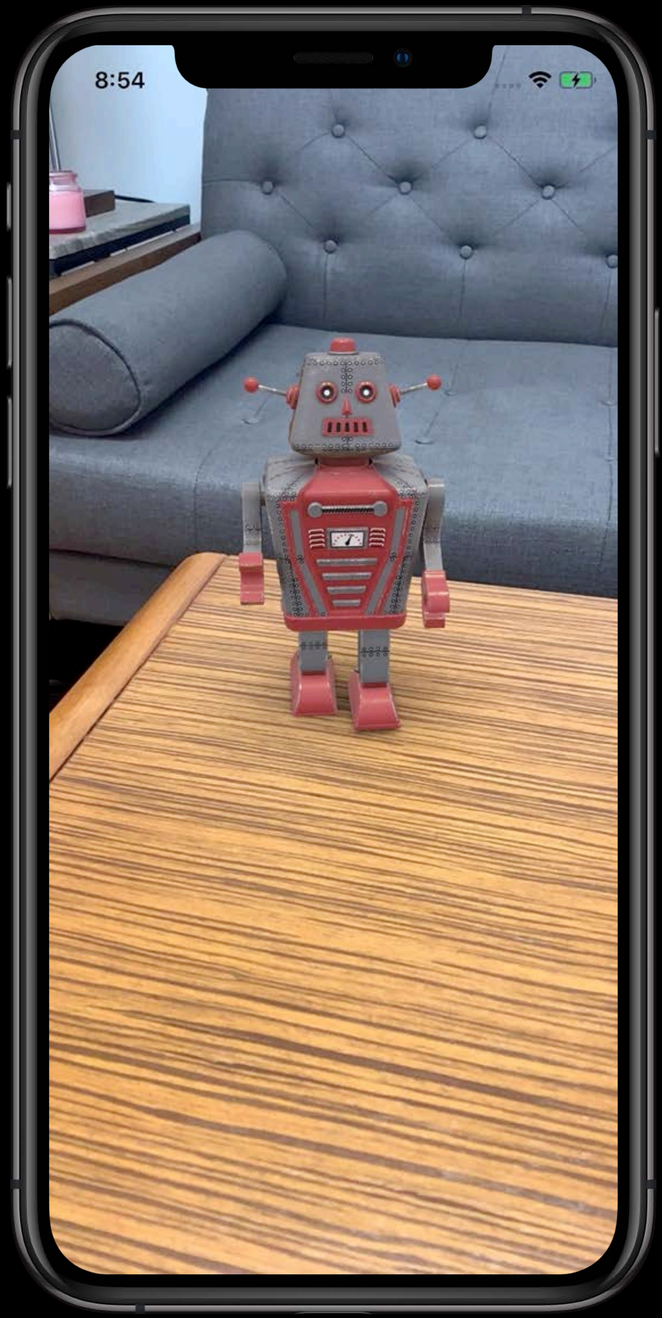


ModelEntity

Represents visual parts of a scene

Load directly from ‘usdz’ or Reality Files

Contains geometry, animation, and physics

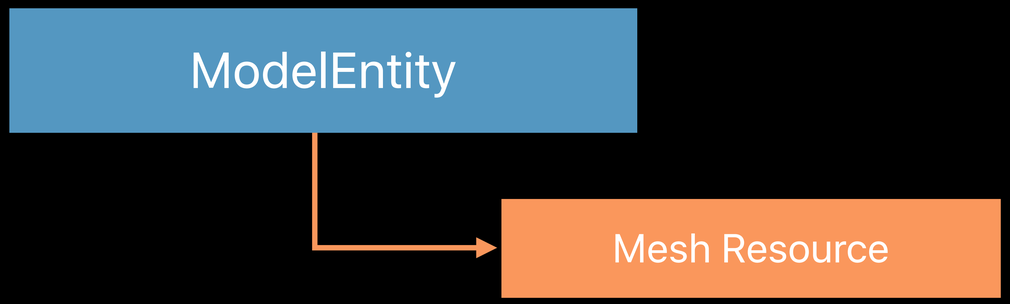


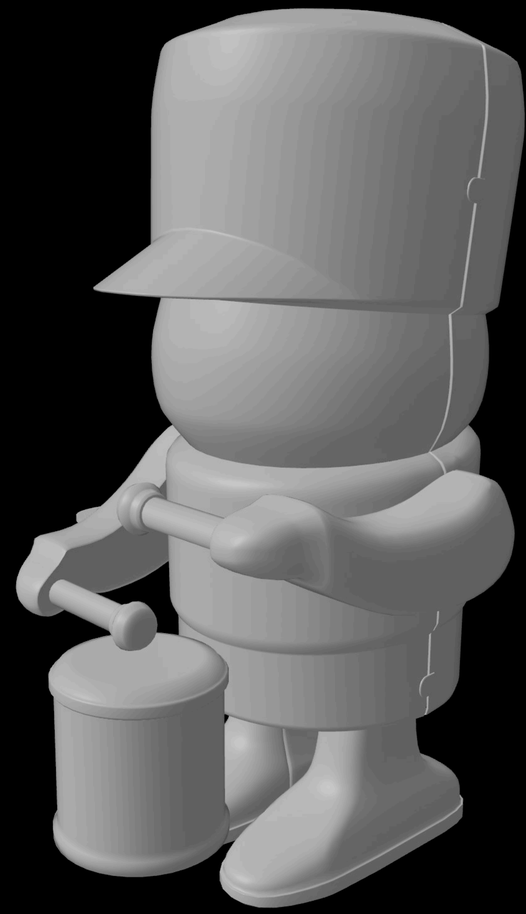
```
/// Create anchor entity for attaching content
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)


/// Load a model entity from file
let robot = try ModelEntity.loadModel(named: "robot")
/// Add model entity to anchor
anchor.addChild(robot)
```

```
/// Create anchor entity for attaching content
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)


/// Load a model entity from file
let robot = try ModelEntity.loadModel(named: "robot")
/// Add model entity to anchor
anchor.addChild(robot)
```

ModelEntity

ModelEntity





Mesh Resource

Provides geometry of scene

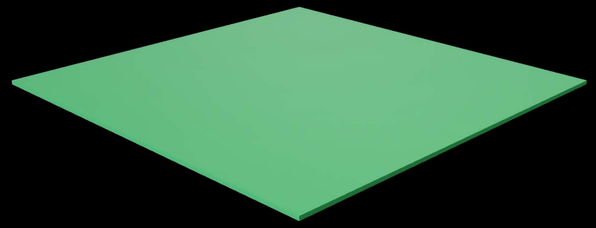
Shared across entities

Built-in primitives

Primitive Meshes

Box

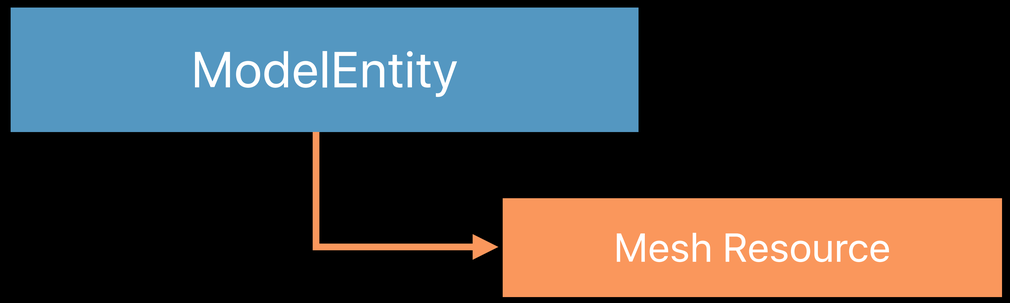
Sphere

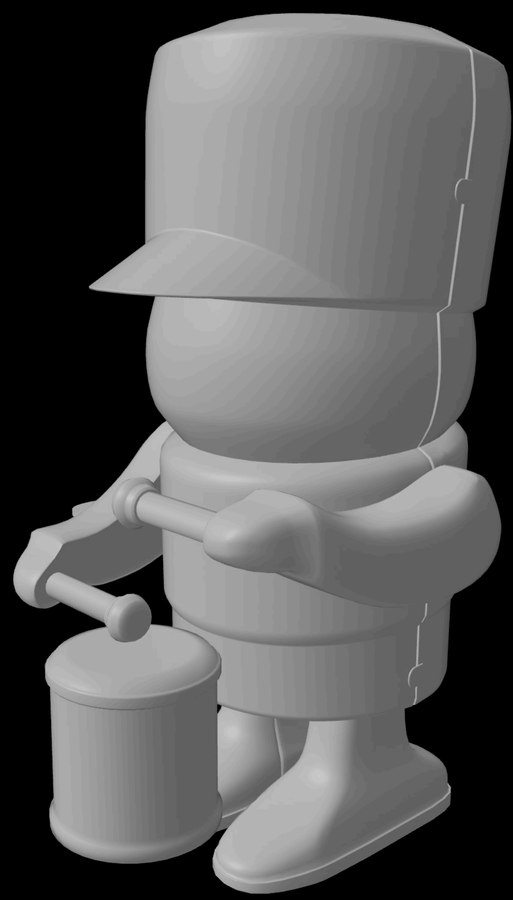


Plane

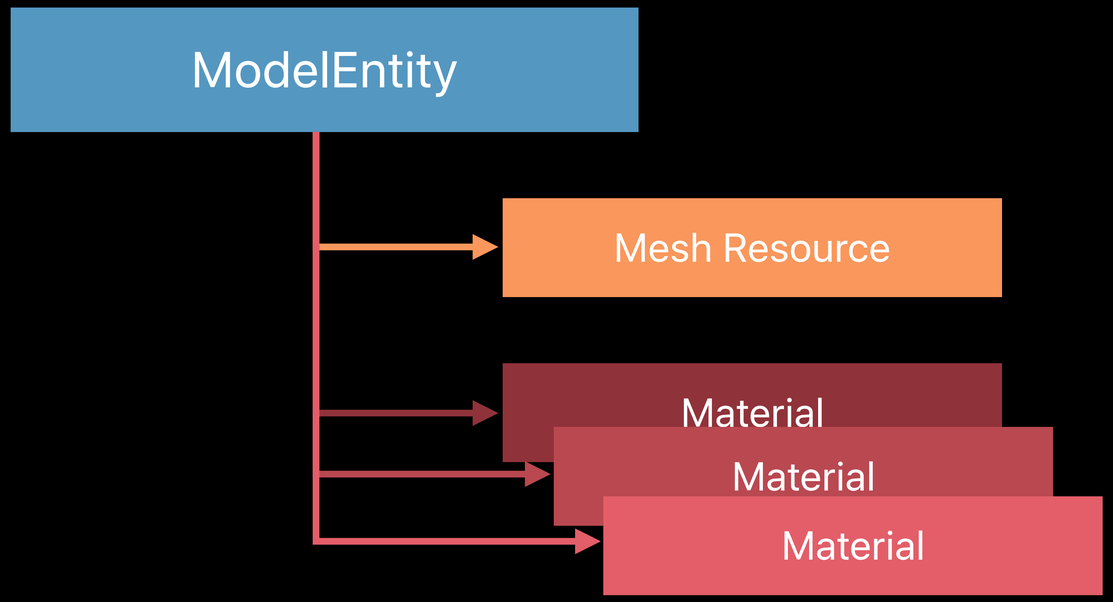
Text

ModelEntity





ModelEntity



Materials

Provides look and feel of an object

Physically based rendering

Load from ‘usdz’ or Reality File

Built-in material types

SimpleMaterial

Physically-based

Easy to use

Supports texture or scalar inputs

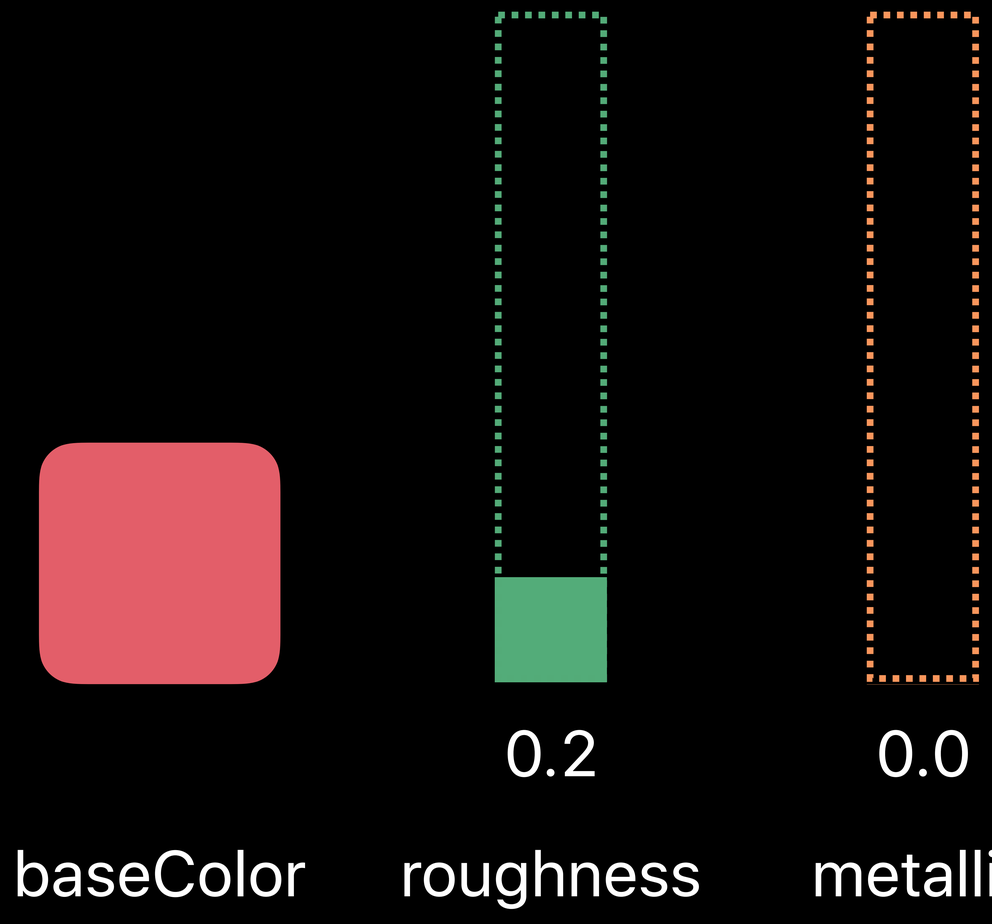
SimpleMaterial

Physically-based

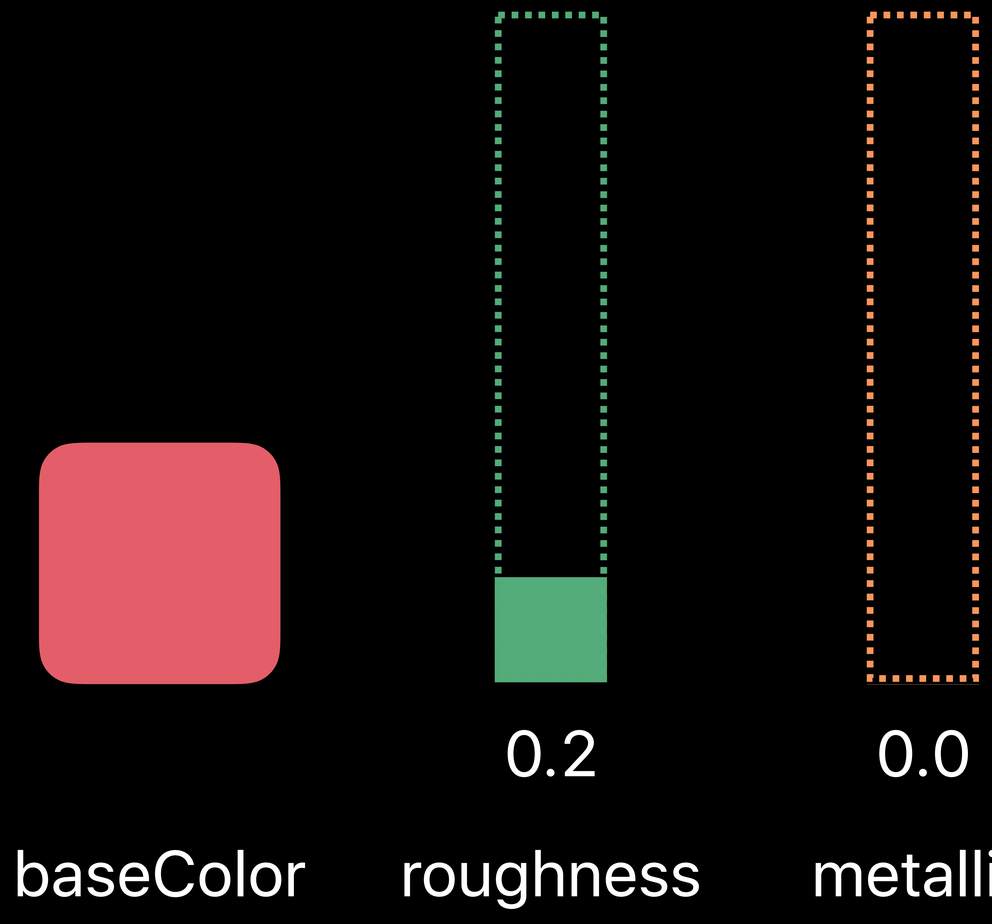
Easy to use

Supports texture or scalar inputs

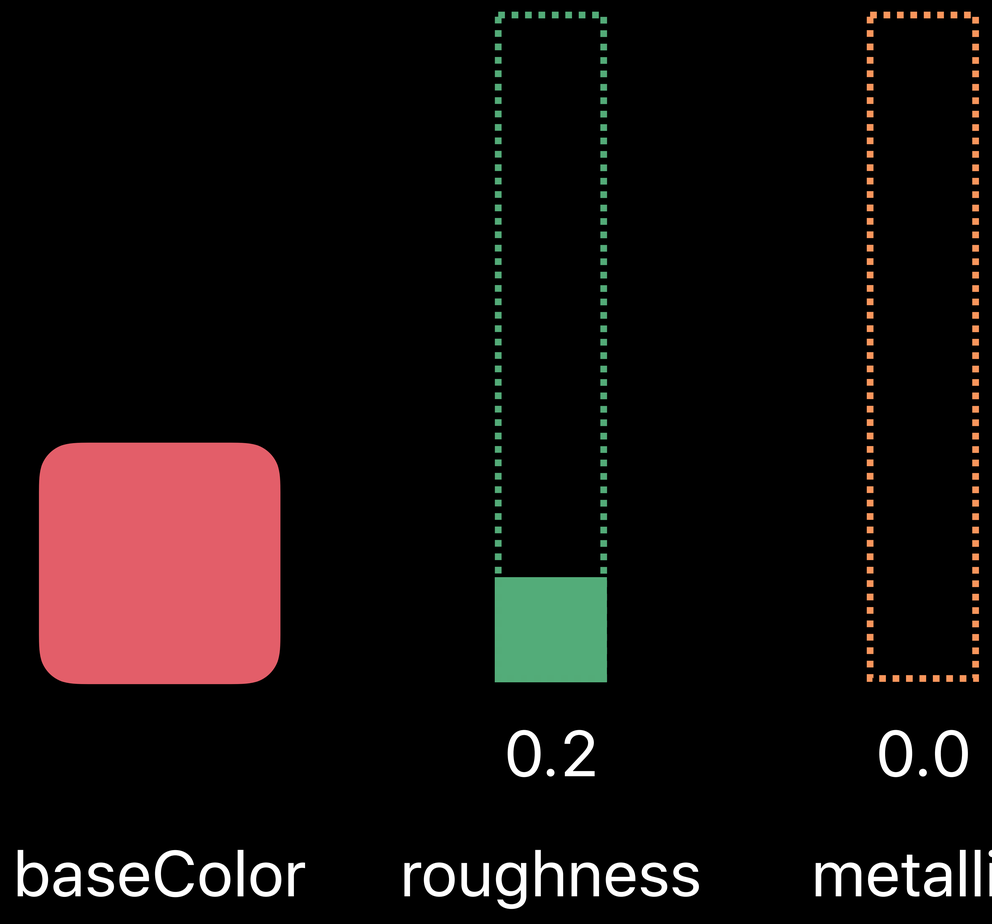
SimpleMaterial



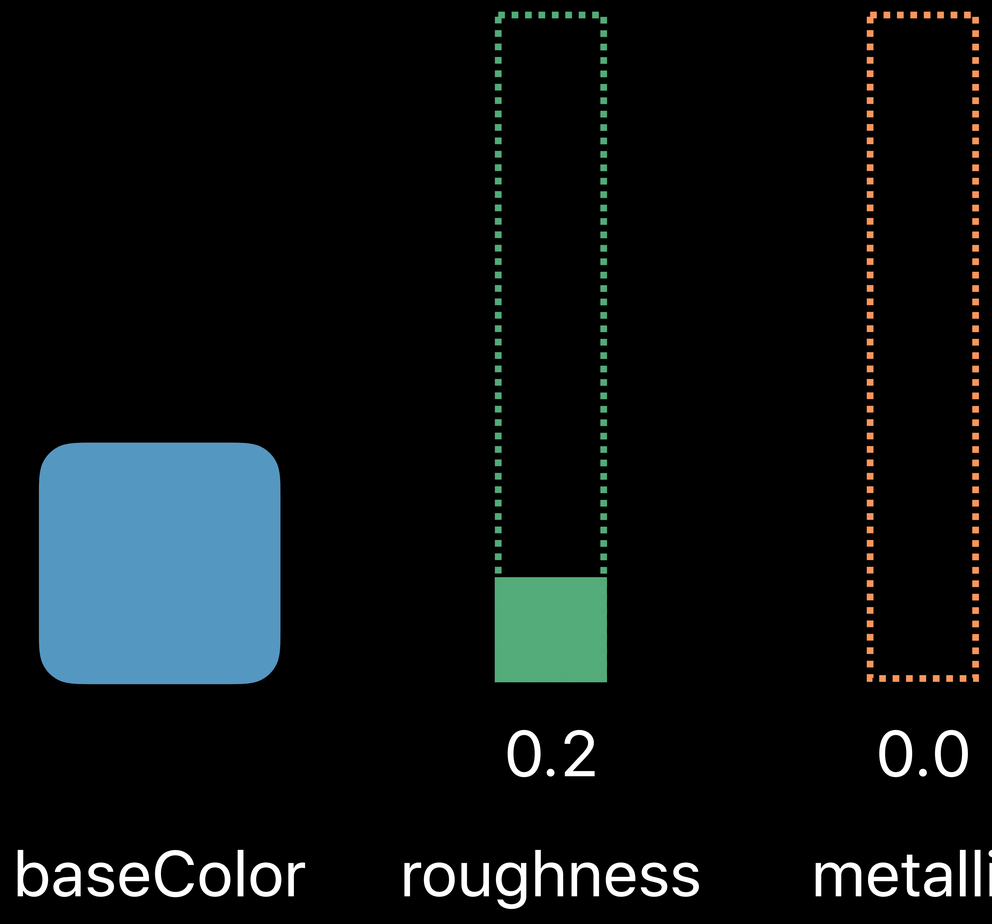
SimpleMaterial



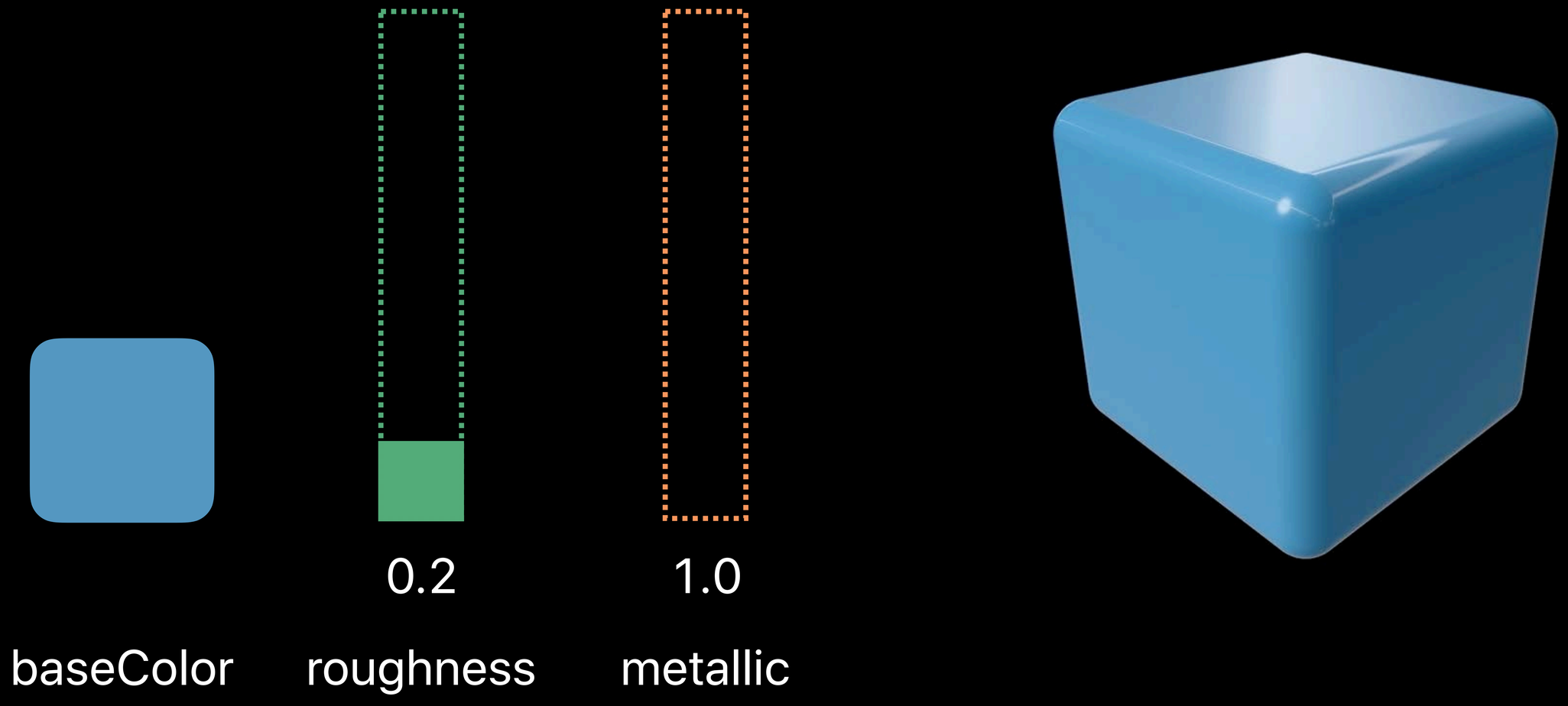
SimpleMaterial



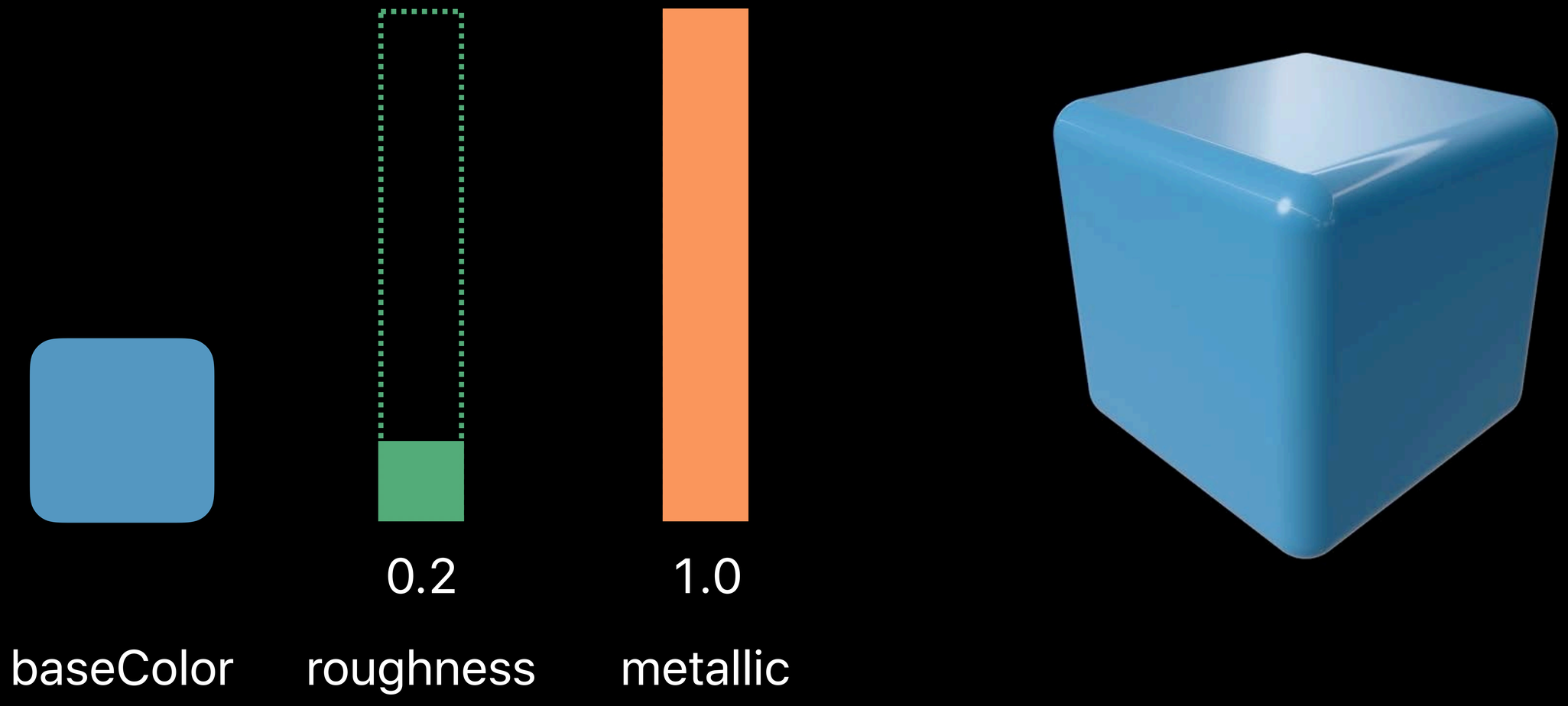
SimpleMaterial



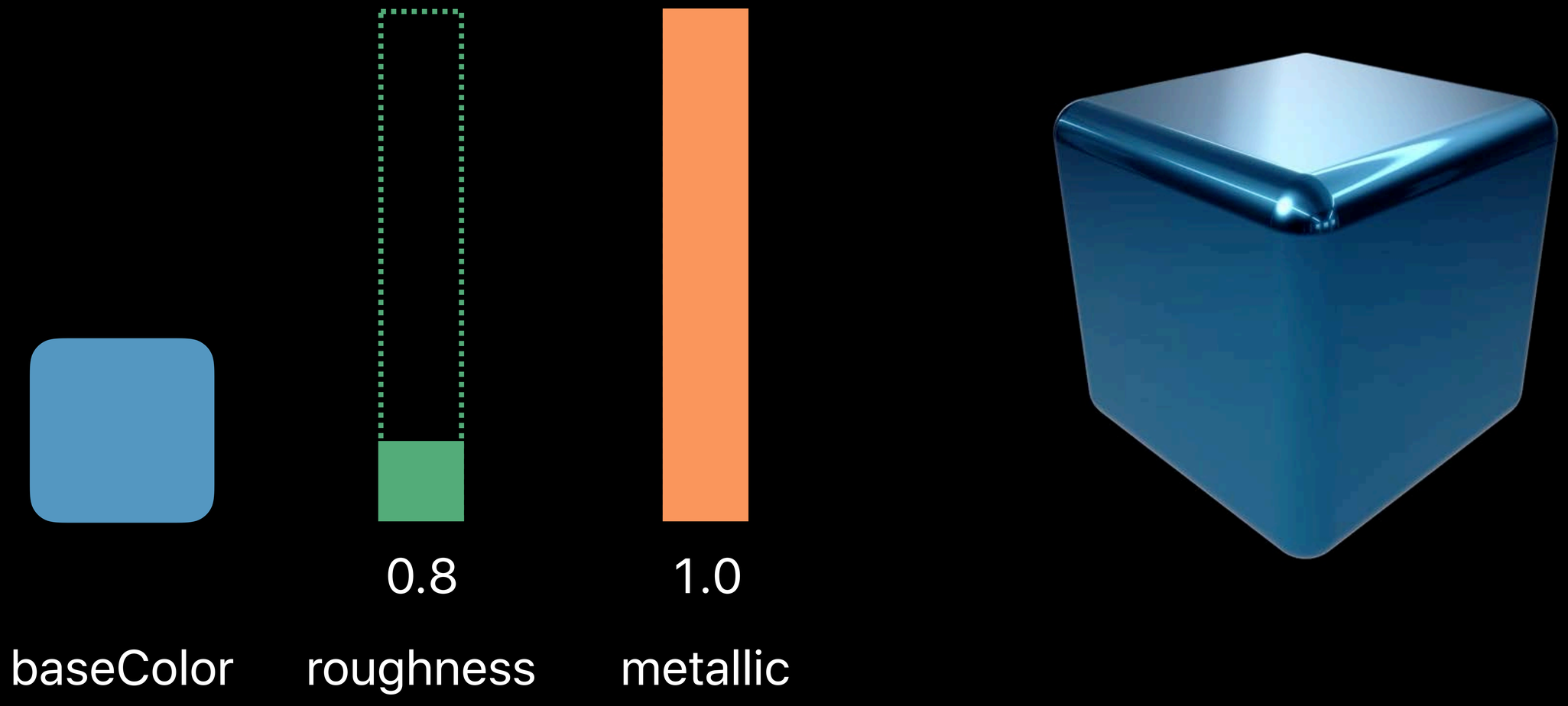
SimpleMaterial



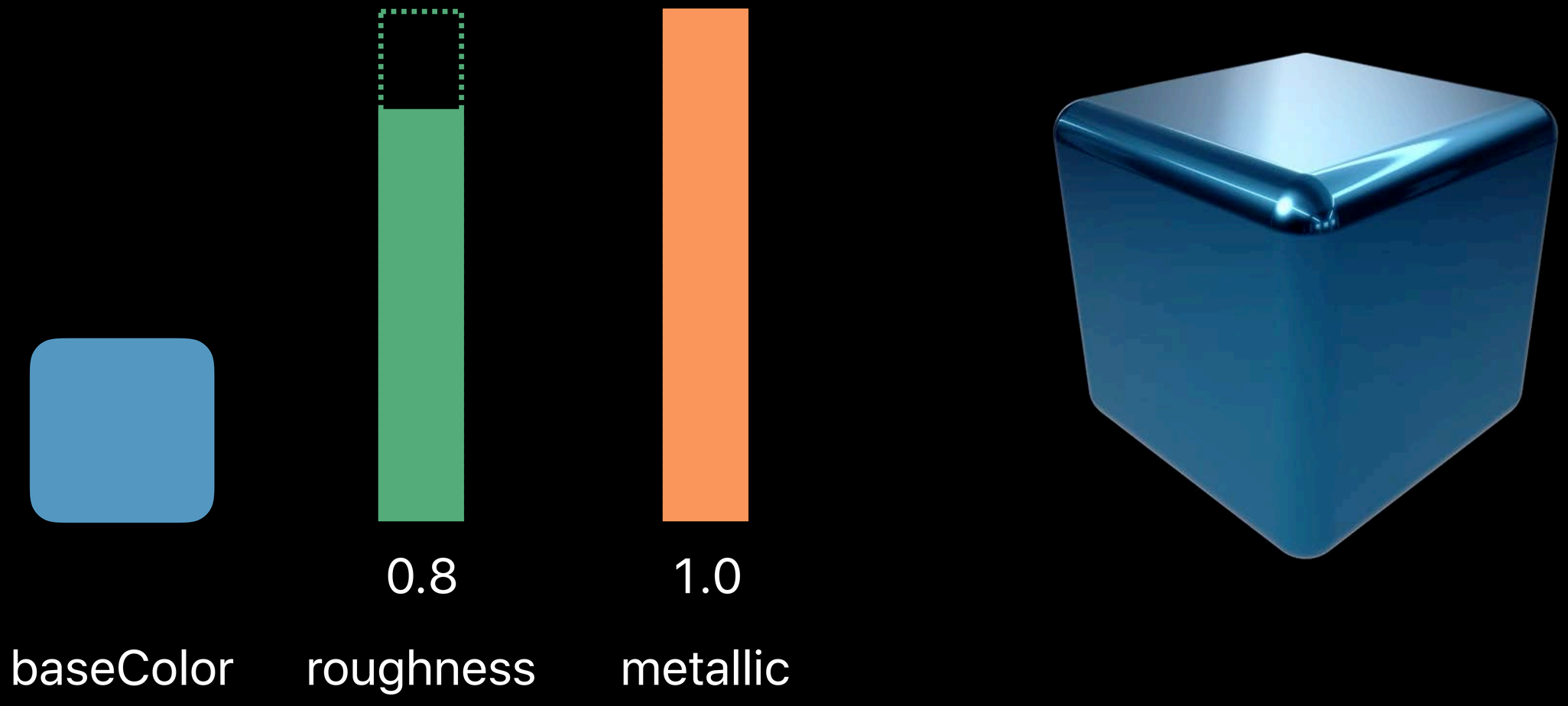
SimpleMaterial



SimpleMaterial



SimpleMaterial

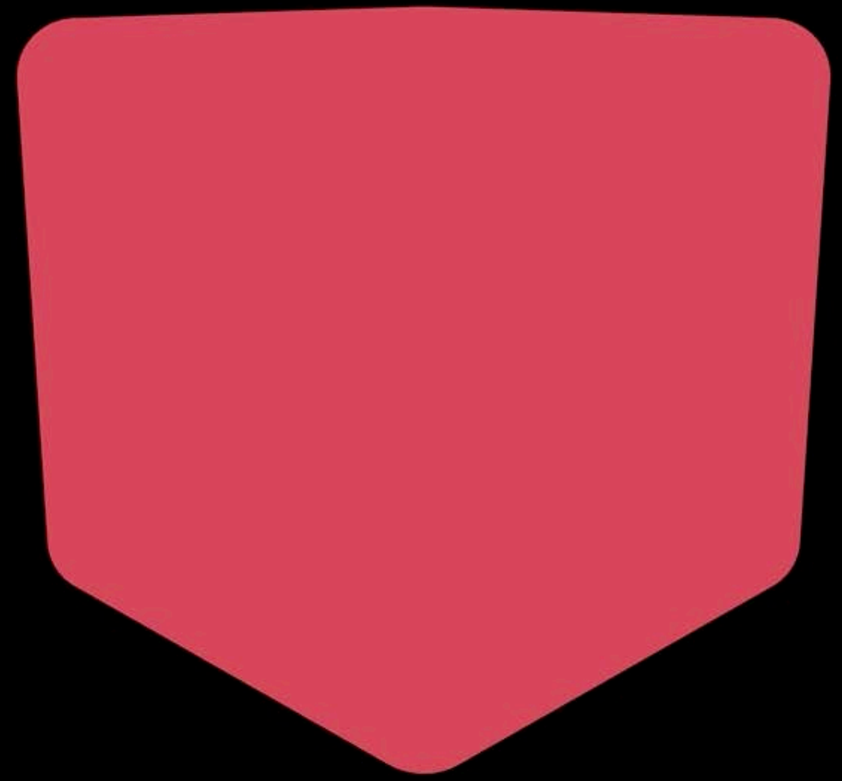


Unlit Material

Provides flat coloring

Useful for emphasized content

Great for debugging

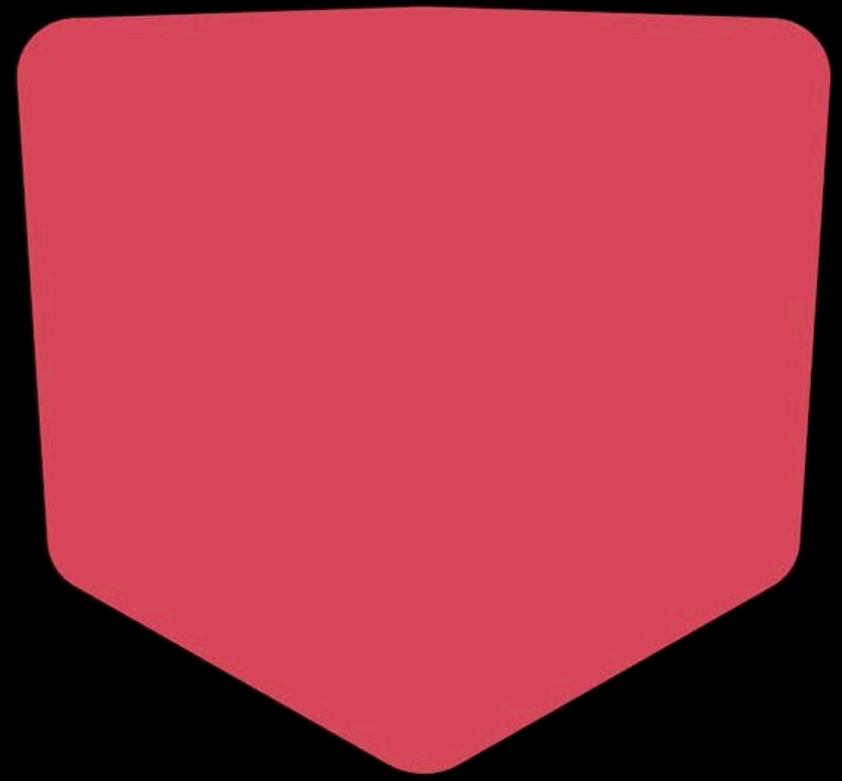


Unlit Material

Provides flat coloring

Useful for emphasized content

Great for debugging



Unlit Material

Provides flat coloring

Useful for emphasized content

Great for debugging

Unlit Material

Provides flat coloring

Useful for emphasized content

Great for debugging

OcclusionMaterial

Reveals video passthrough

Simulates real world objects

Can receive dynamic lighting

OcclusionMaterial

Reveals video passthrough

Simulates real world objects

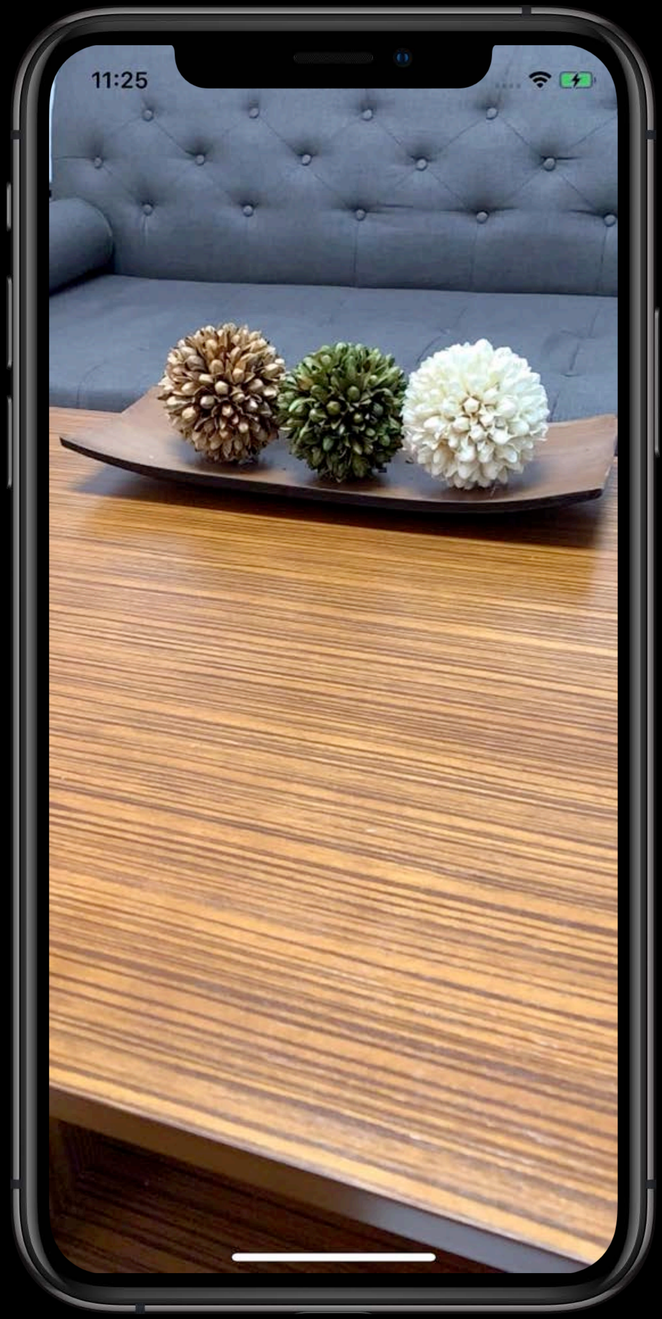
Can receive dynamic lighting

OcclusionMaterial

Reveals video passthrough

Simulates real world objects

Can receive dynamic lighting

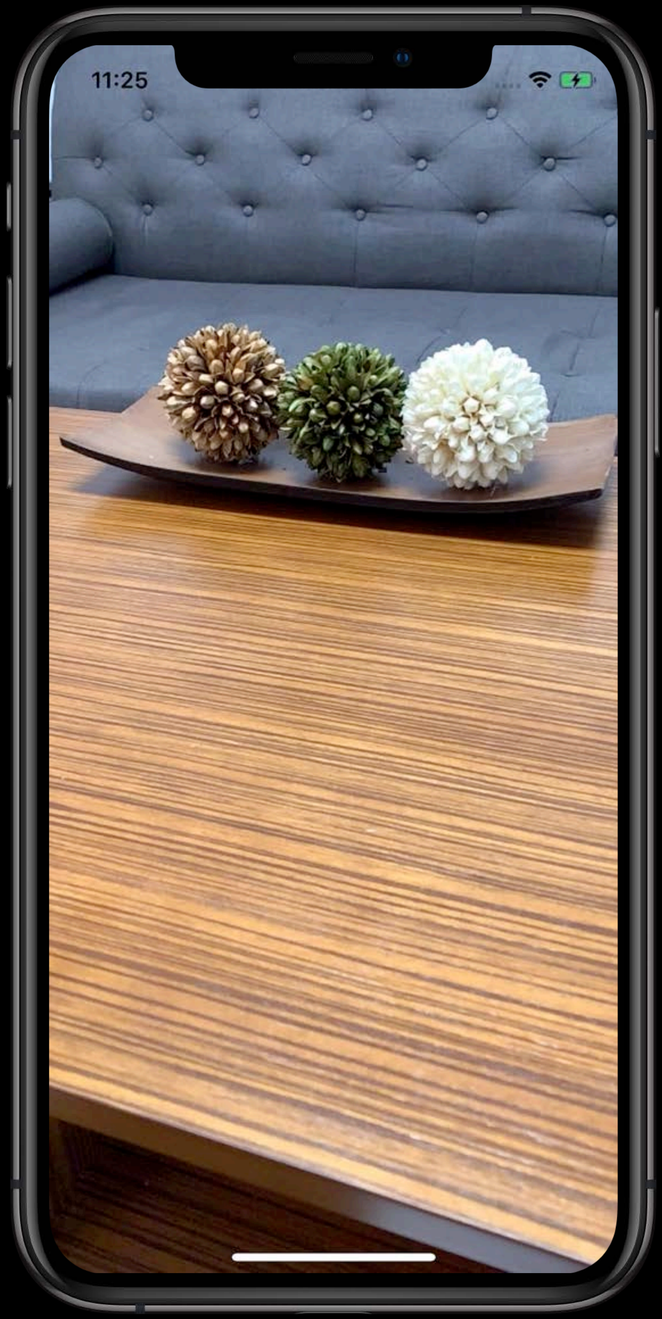


OcclusionMaterial

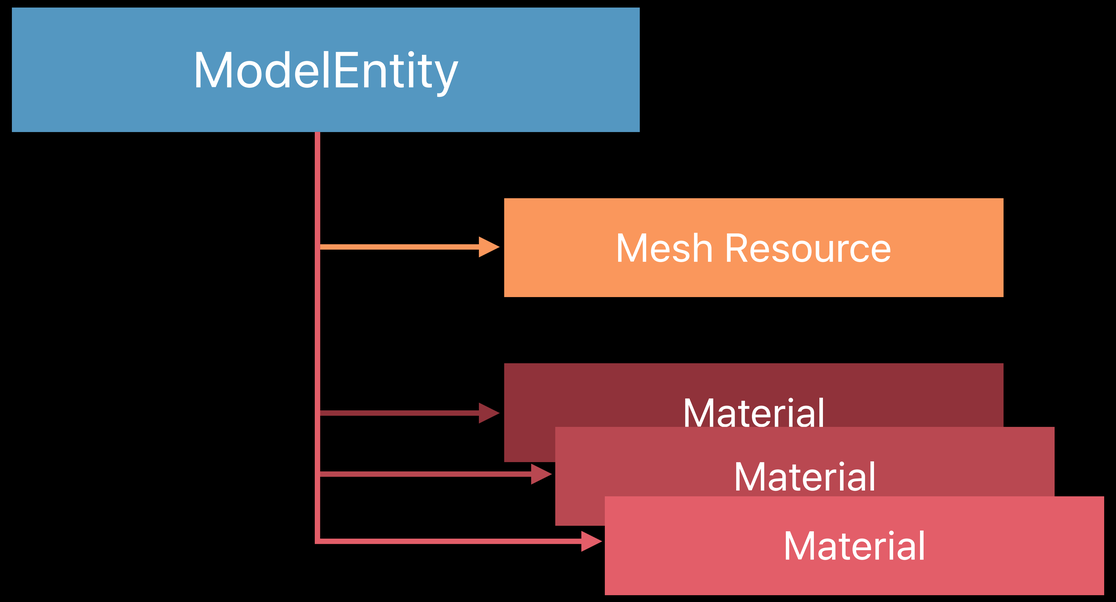
Reveals video passthrough

Simulates real world objects

Can receive dynamic lighting



ModelEntity



```
/// Create anchor entity for attaching content
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)

/// Generate a box mesh
let box = MeshResource.generateBox(size: 1.0 , cornerRadius: 0.1)
/// Create a simple metallic gray material
let metal = SimpleMaterial(color: .gray, isMetallic: true)

/// Create a model entity from mesh and material
let model = try ModelEntity(mesh: box, materials: [metal])
anchor.addChild(model)
```

```
/// Create anchor entity for attaching content
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)

/// Generate a box mesh
let box = MeshResource.generateBox(size: 1.0 , cornerRadius: 0.1)
/// Create a simple metallic gray material
let metal = SimpleMaterial(color: .gray, isMetallic: true)

/// Create a model entity from mesh and material
let model = try ModelEntity(mesh: box, materials: [metal])
anchor.addChild(model)
```

```
/// Create anchor entity for attaching content
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)

/// Generate a box mesh
let box = MeshResource.generateBox(size: 1.0 , cornerRadius: 0.1)
/// Create a simple metallic gray material
let metal = SimpleMaterial(color: .gray, isMetallic: true)

/// Create a model entity from mesh and material
let model = try ModelEntity(mesh: box, materials: [metal])
anchor.addChild(model)
```

Animation

Supports skeletal and transform

Load from ‘usdz’ or Reality File

```
/// Start playing animation
let controller = entity.playAnimation(named: "dance")

/// Pause animation
controller.pause()

if controller.isPaused {
    /// Resume animation
    controller.resume()
}

/// Stop animation
controller.stop()
```

```
/// Start playing animation
let controller = entity.playAnimation(named: "dance")

/// Pause animation
controller.pause()

if controller.isPaused {
    /// Resume animation
    controller.resume()
}

/// Stop animation
controller.stop()
```

```swift
/// Start playing animation
let controller = entity.playAnimation(named: "dance")

/// Pause animation
controller.pause()

if controller.isPaused {
    /// Resume animation
    controller.resume()
}

/// Stop animation
controller.stop()
```

```swift
/// Start playing animation
let controller = entity.playAnimation(named: "dance")

/// Pause animation
controller.pause()

if controller.isPaused {
    /// Resume animation
    controller.resume()
}

/// Stop animation
controller.stop()
```

```
/// Start playing animation
let controller = entity.playAnimation(named: "dance")

/// Pause animation
controller.pause()

if controller.isPaused {
    /// Resume animation
    controller.resume()
}

/// Stop animation
controller.stop()
```

```
let destination = Transform(translation: [0.0, 0.0, 5.0]) /// Move forward by 5 meters

let controller = entity.move(to: destination,
                             relativeTo: nil,    /// in world space
                             duration: 1.0,      /// for 2 seconds
                             easing: .easeInOut) /// easing in and out
```

```
let destination = Transform(translation: [0.0, 0.0, 5.0]) /// Move forward by 5 meters

let controller = entity.move(to: destination,
                             relativeTo: nil,    /// in world space
                             duration: 1.0,      /// for 2 seconds
                             easing: .easeInOut) /// easing in and out
```

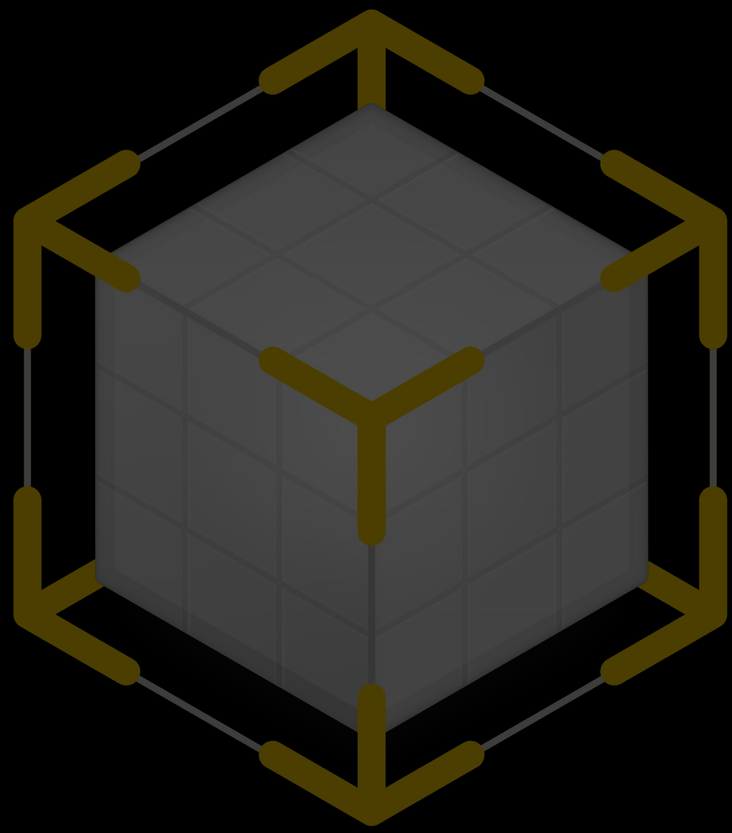
```
let destination = Transform(translation: [0.0, 0.0, 5.0]) /// Move forward by 5 meters

let controller = entity.move(to: destination,
                             relativeTo: nil,    /// in world space
                             duration: 1.0,      /// for 2 seconds
                             easing: .easeInOut) /// easing in and out
```

```
let destination = Transform(translation: [0.0, 0.0, 5.0]) /// Move forward by 5 meters

let controller = entity.move(to: destination,
                             relativeTo: nil,    /// in world space
                             duration: 1.0,      /// for 2 seconds
                             easing: .easeInOut) /// easing in and out
```

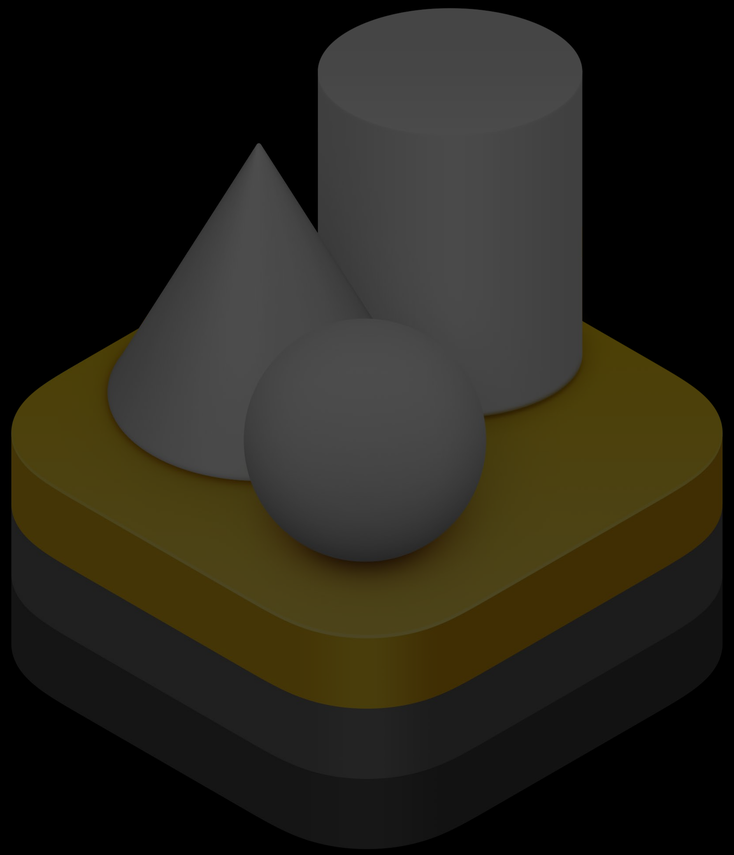
```
let destination = Transform(translation: [0.0, 0.0, 5.0]) /// Move forward by 5 meters

let controller = entity.move(to: destination,
                             relativeTo: nil,    /// in world space
                             duration: 1.0,      /// for 2 seconds
                             easing: .easeInOut) /// easing in and out
```

```
let destination = Transform(translation: [0.0, 0.0, 5.0]) /// Move forward by 5 meters

let controller = entity.move(to: destination,
                             relativeTo: nil,    /// in world space
                             duration: 1.0,      /// for 2 seconds
                             easing: .easeInOut) /// easing in and out
```

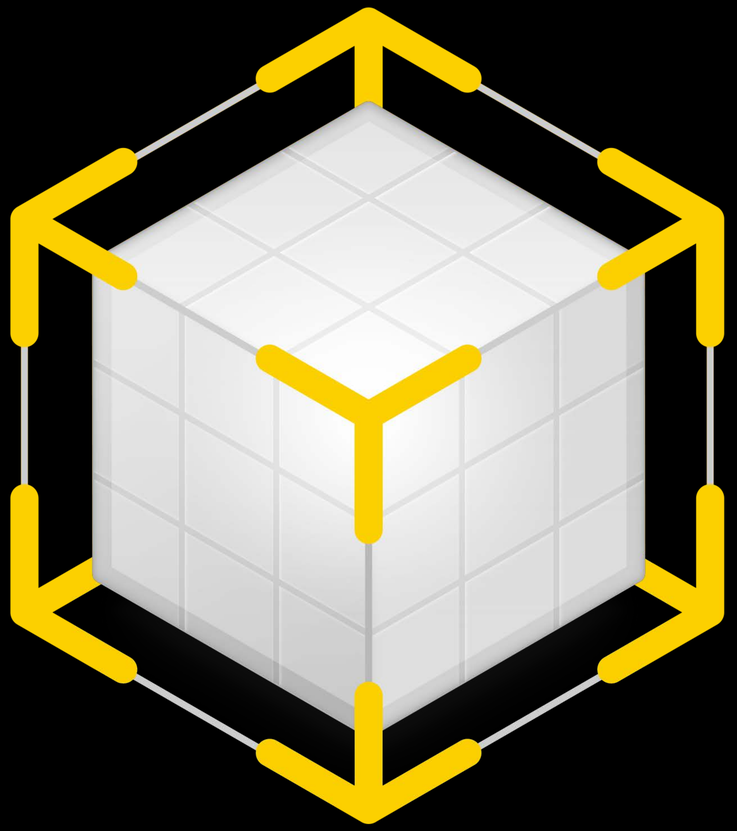
RealityKit



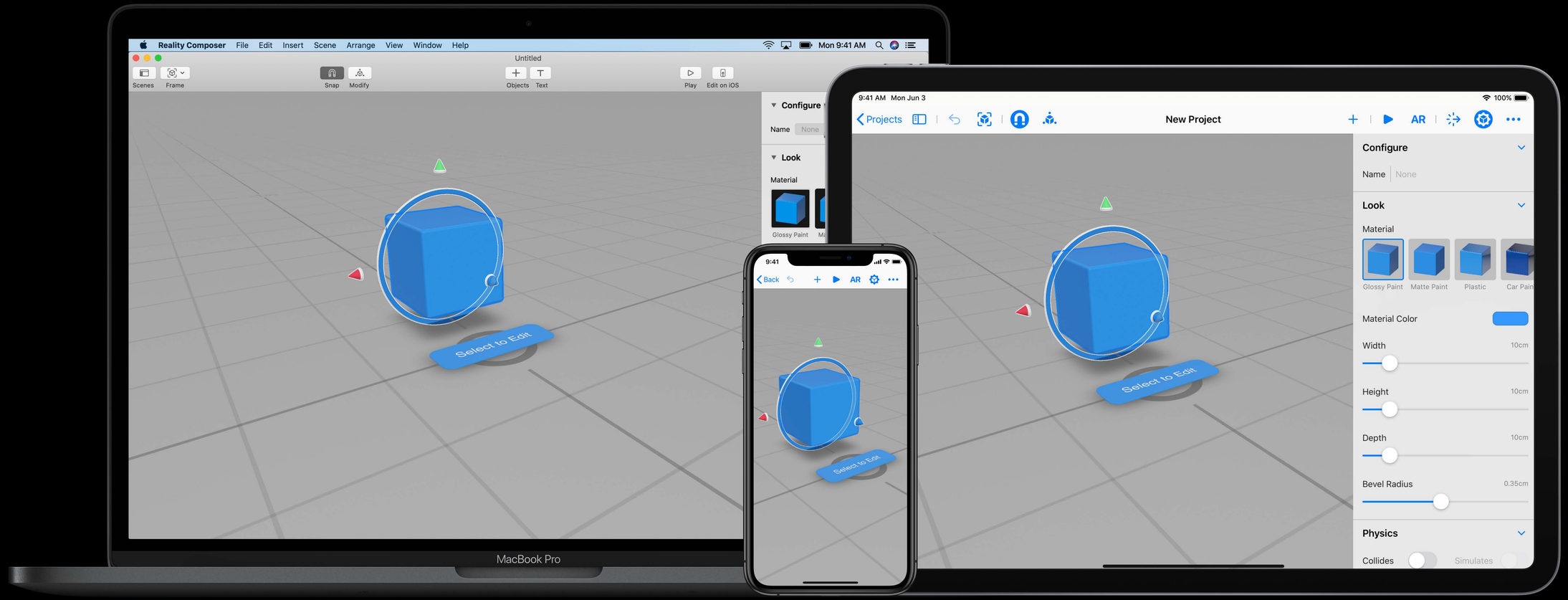
Reality Composer



RealityKit



Reality Composer


Reality Composer File Edit Insert Scene Arrange View Window Help
Mon 9:41 AM
Untitled
Scenes
Frame
Snap
Modify
Objects
Text
Play
Edit on iOS
Configure
Name
None
Look
Material
Glossy Paint
Select to Edit
MacBook Pro
9:41
Back
AR
9:41 AM Mon Jun 3
100%
Projects
New Project
AR
Configure
Name
None
Look
Material
Glossy Paint
Matte Paint
Plastic
Car Paint
Material Color
Width
10cm
Height
10cm
Depth
10cm
Bevel Radius
0.35cm
Physics
Collides
Simulates

Reality Composer

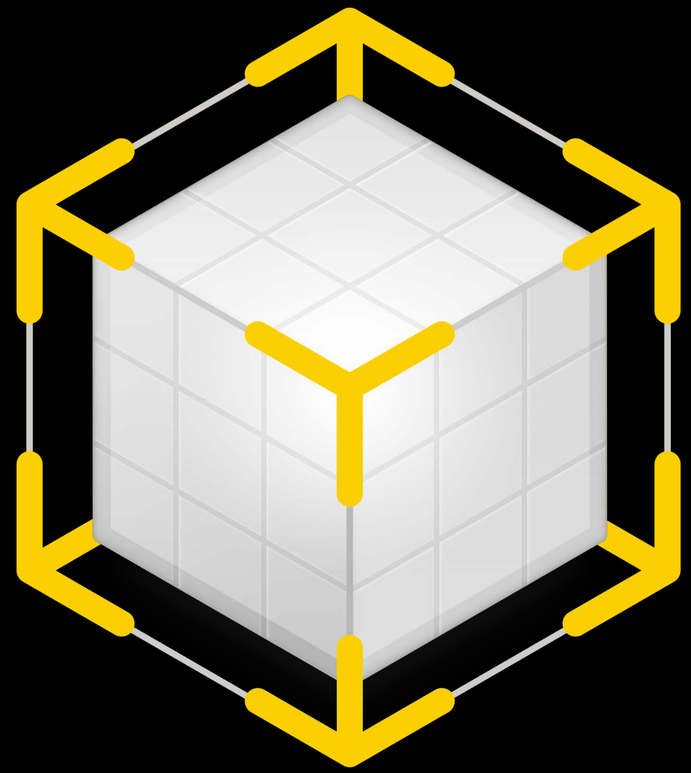
Get started with AR and 3D

Content library

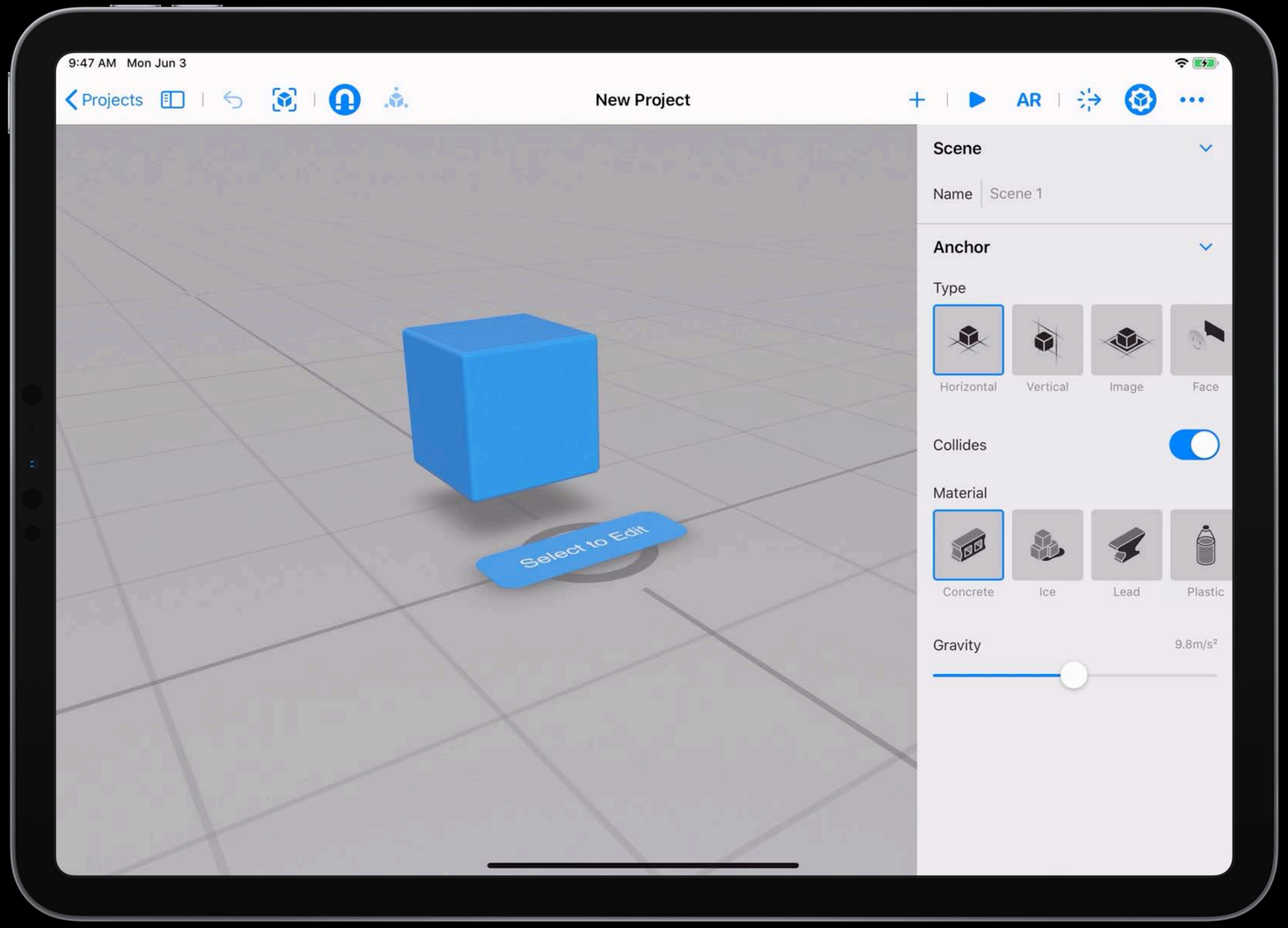
Layout and pre-visualization

Simple interactions

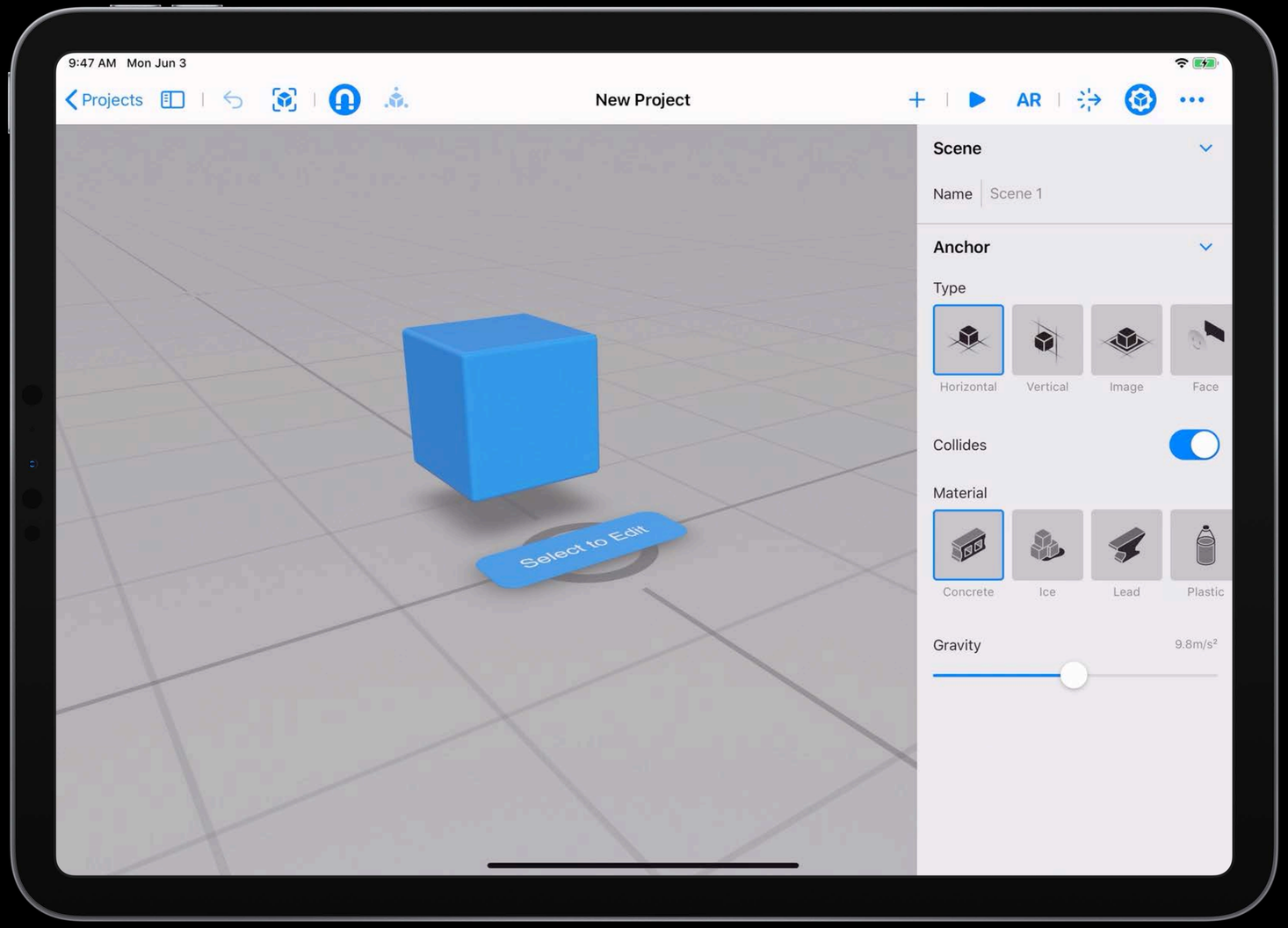
Xcode integration



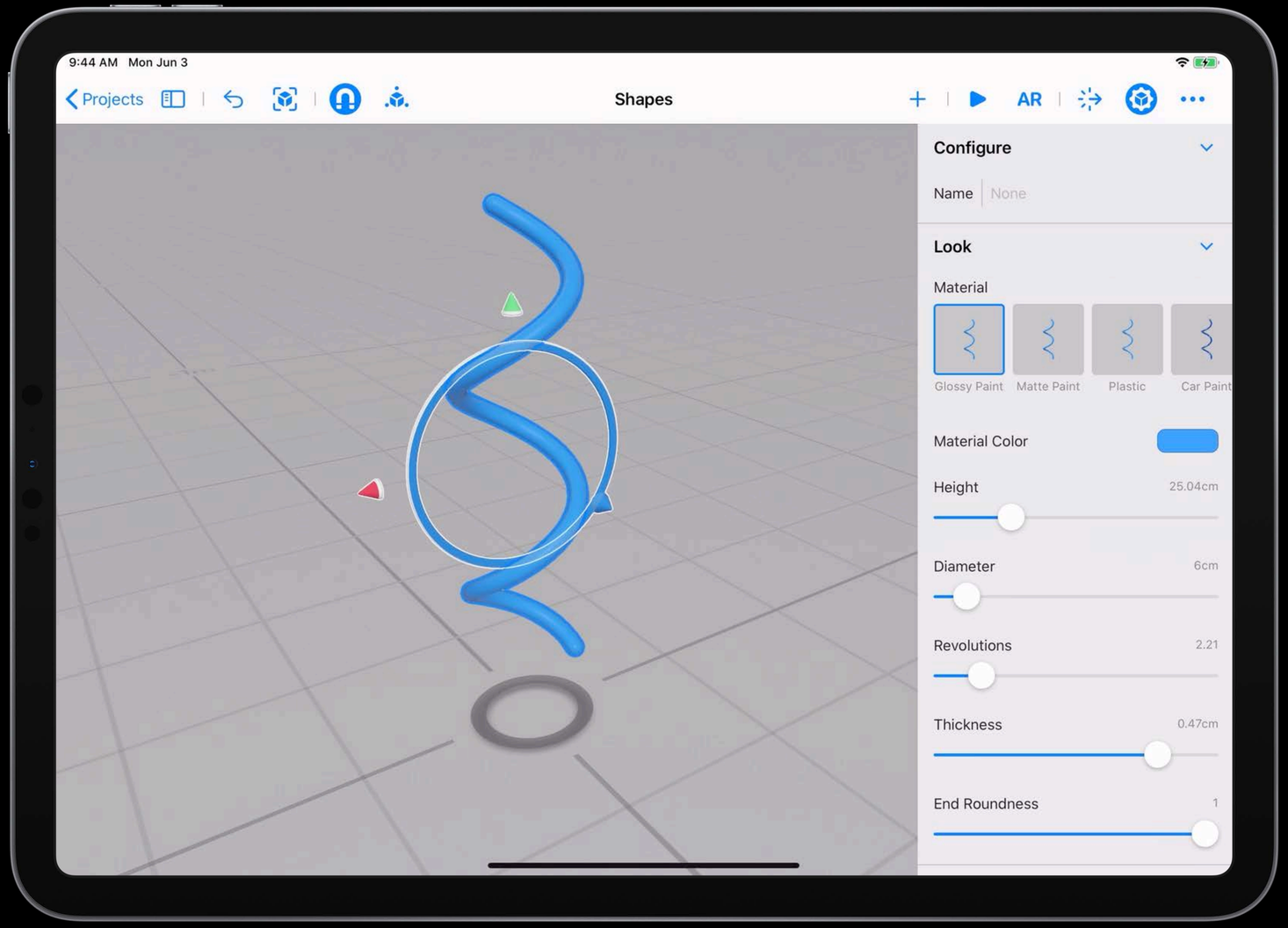
Content Library



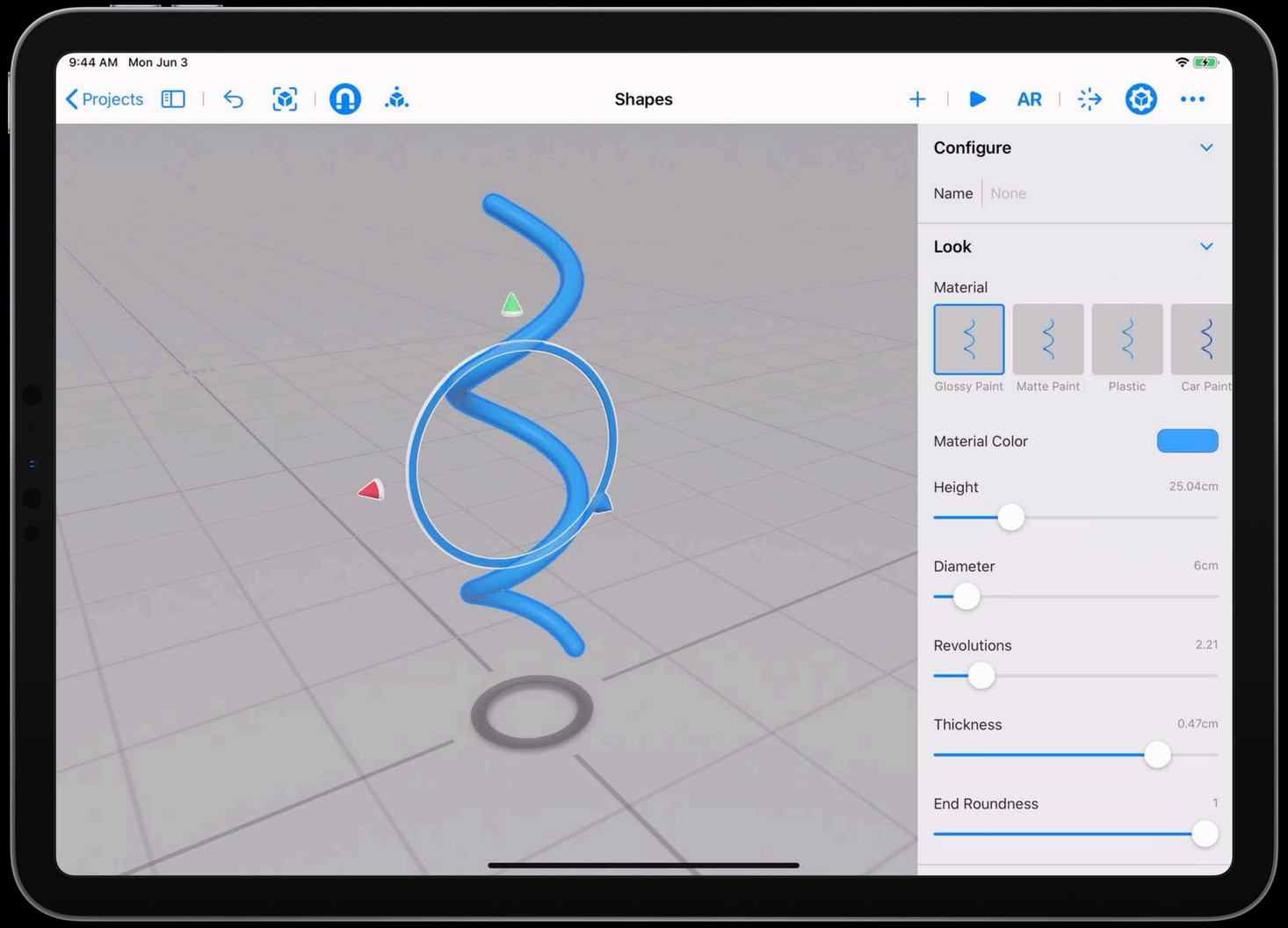
Content Library



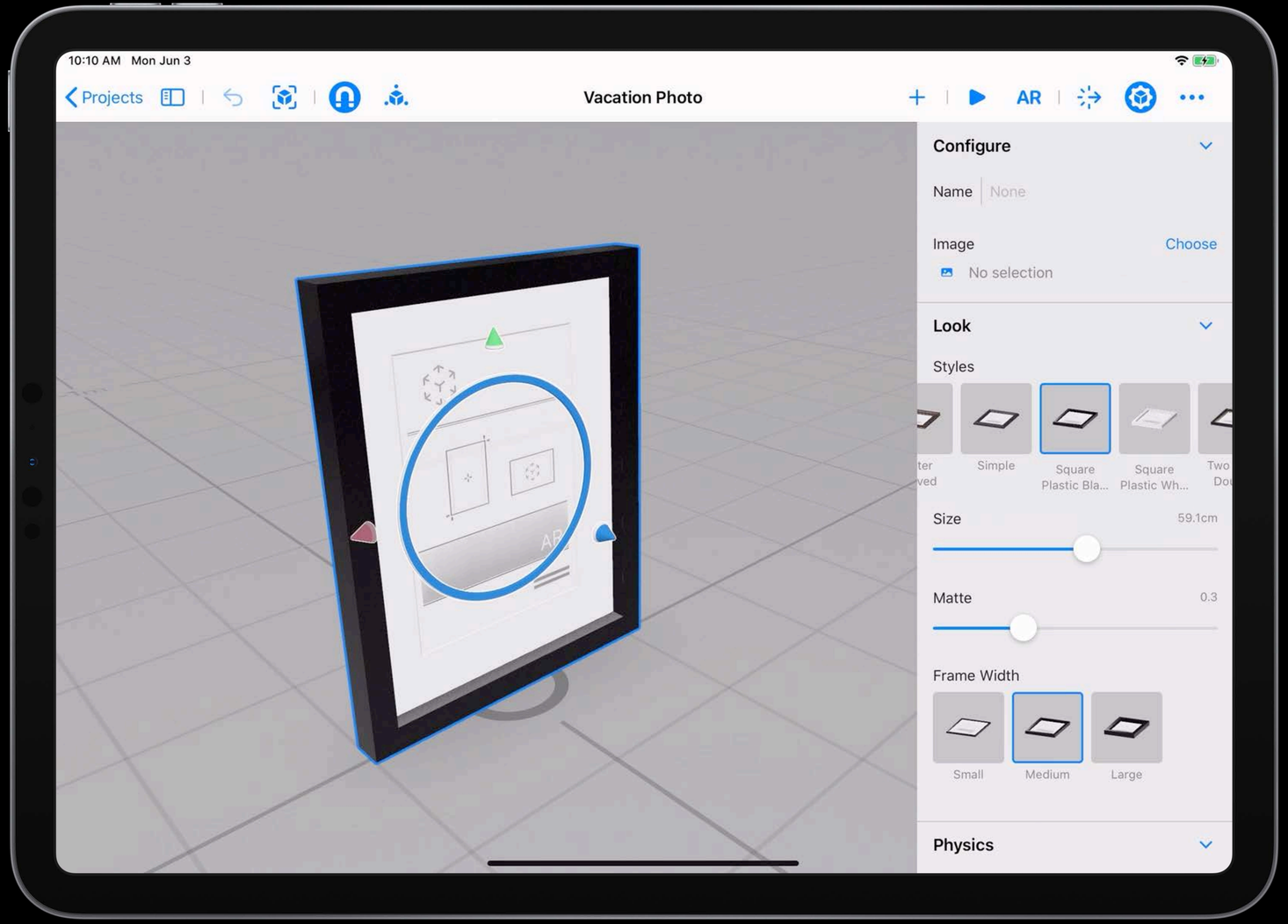
Content Library



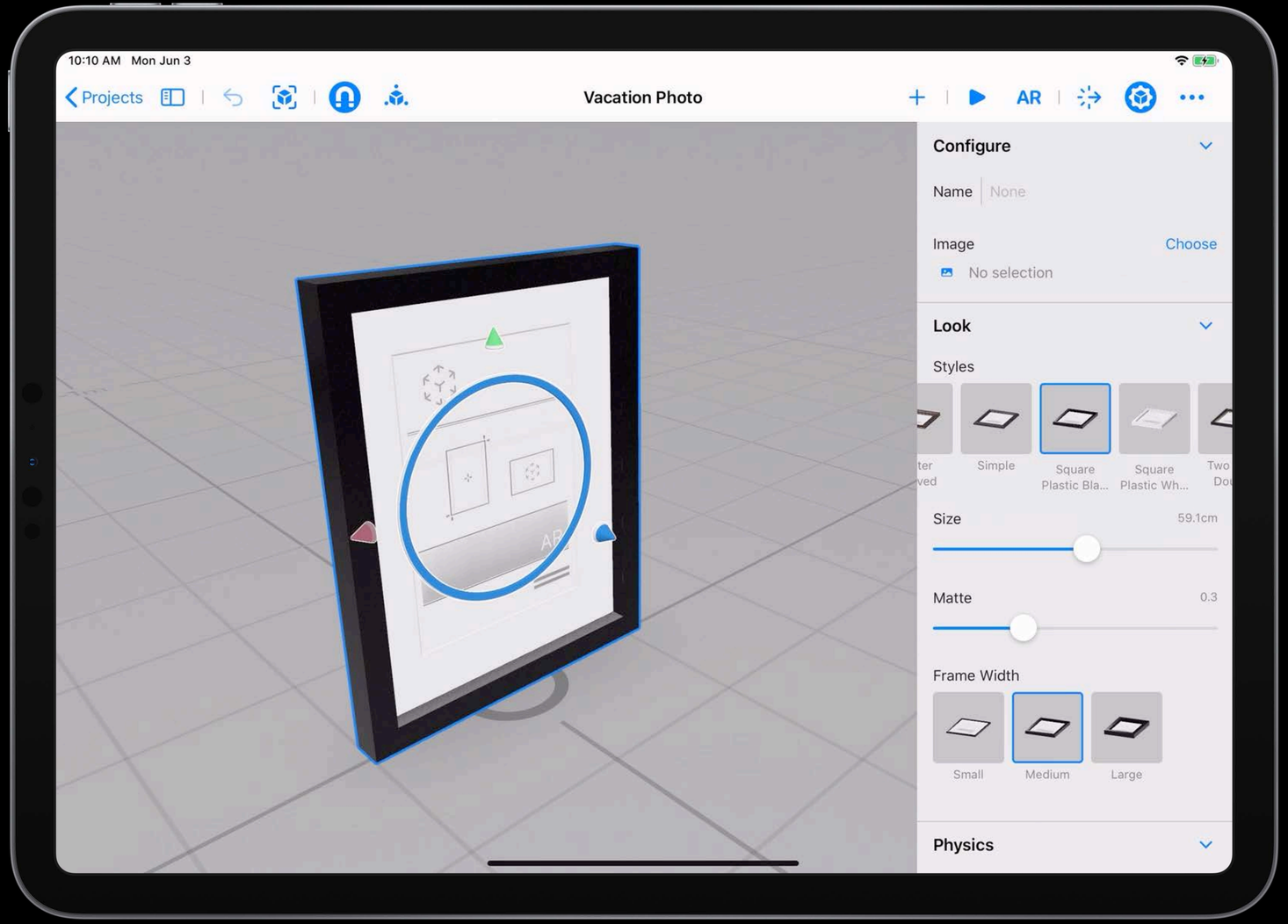
Content Library



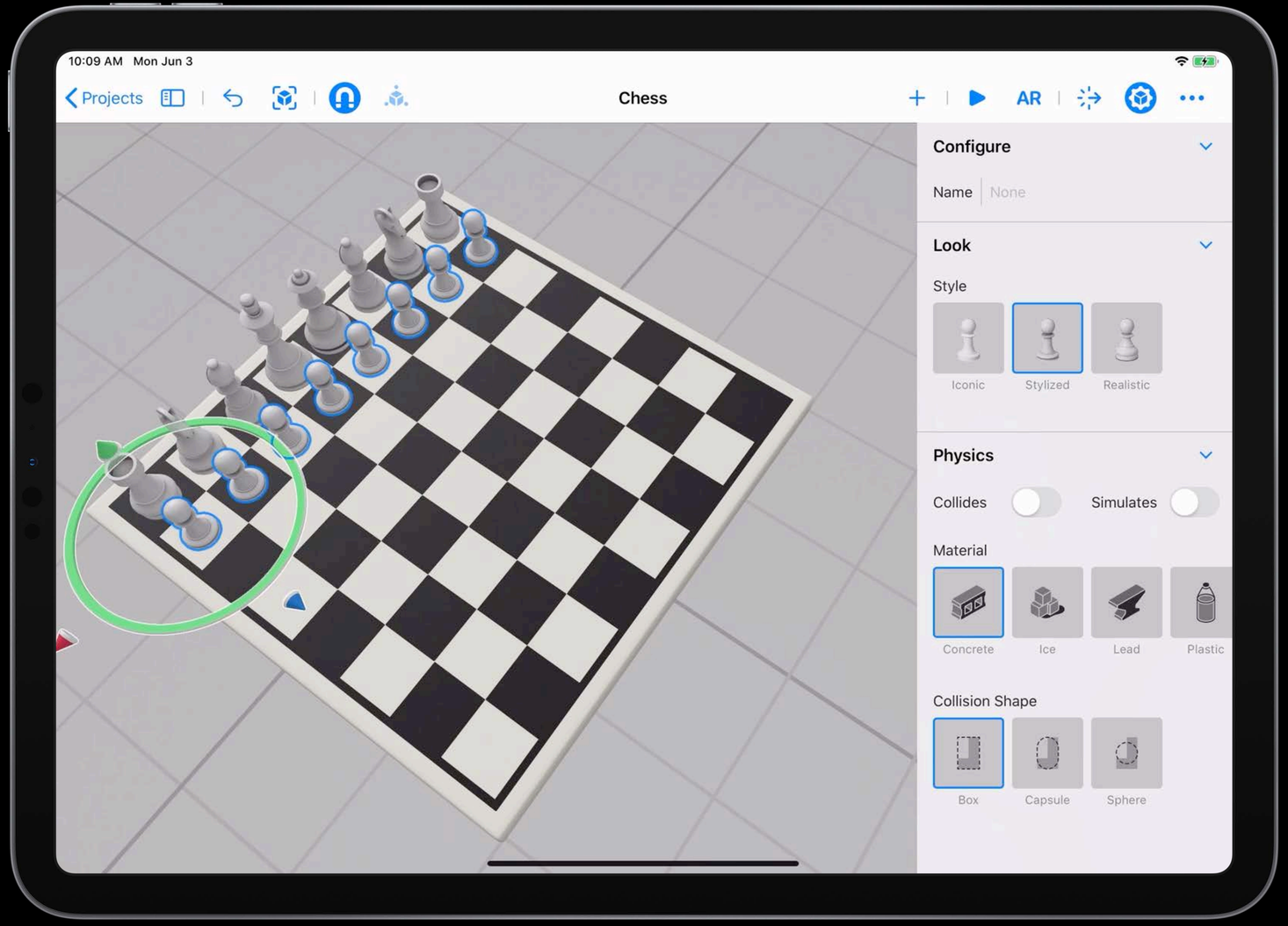
Content Library



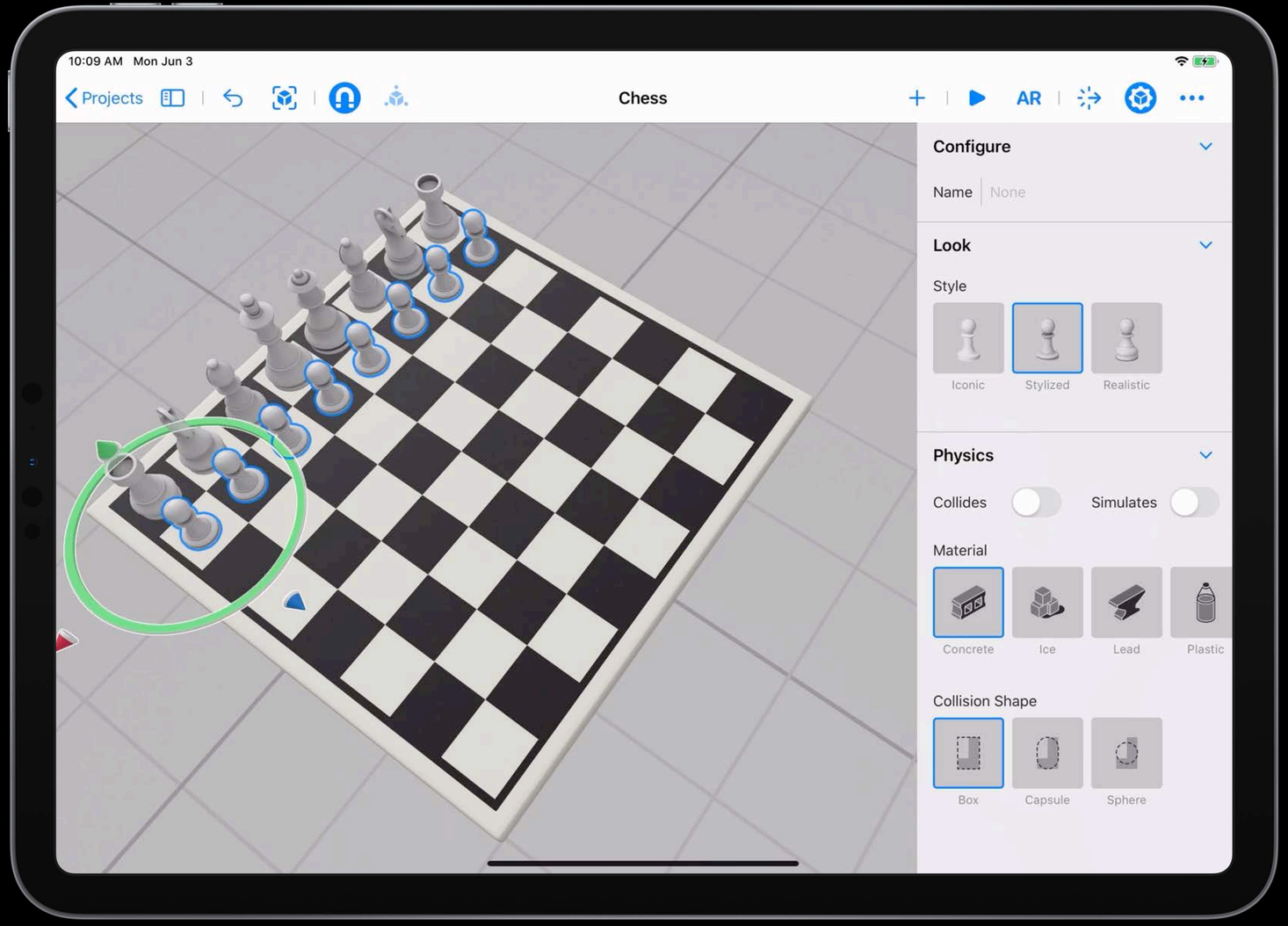
Content Library



Layout and Pre-Visualization



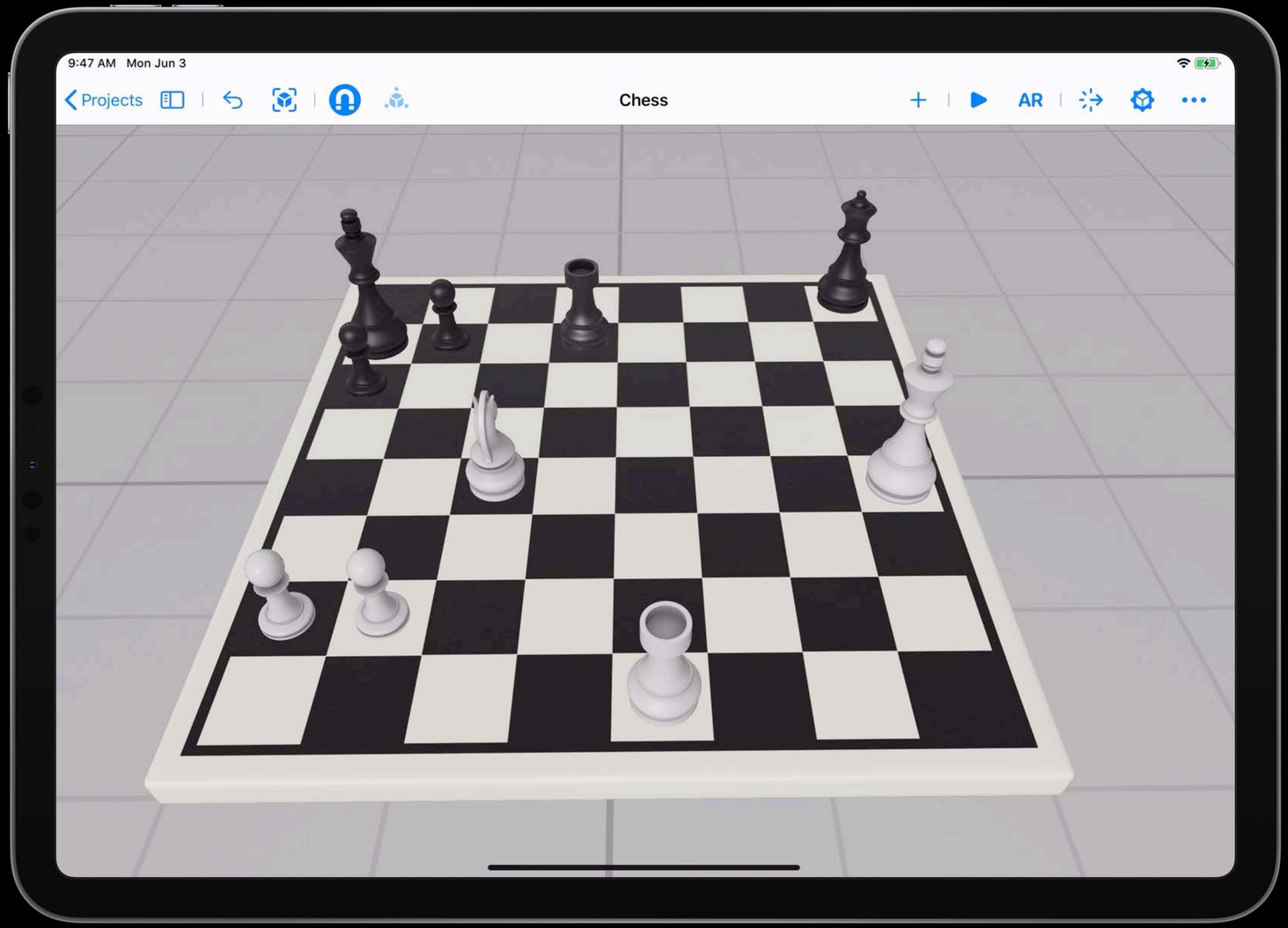
Layout and Pre-Visualization



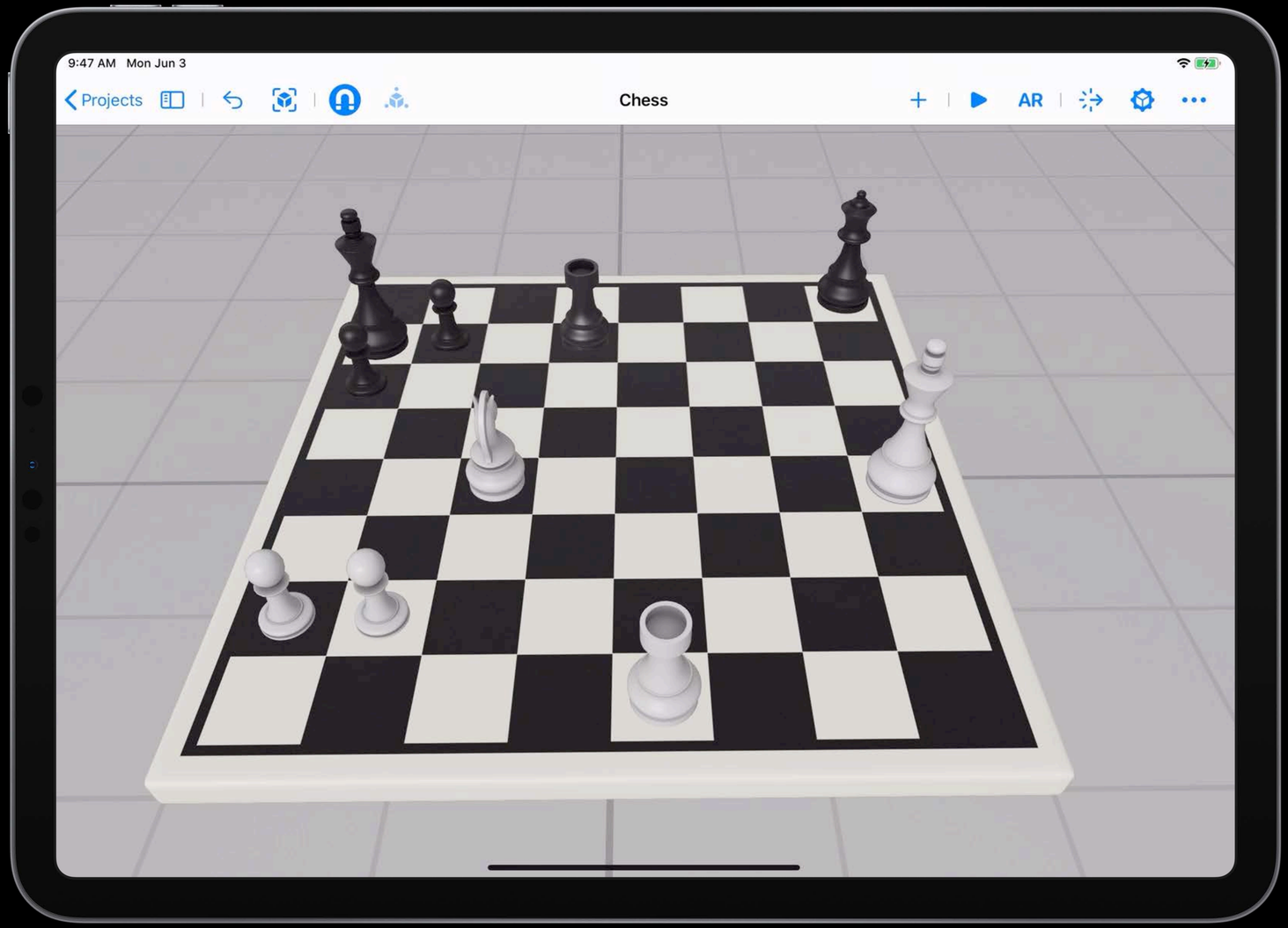
Layout and Pre-Visualization

Layout and Pre-Visualization

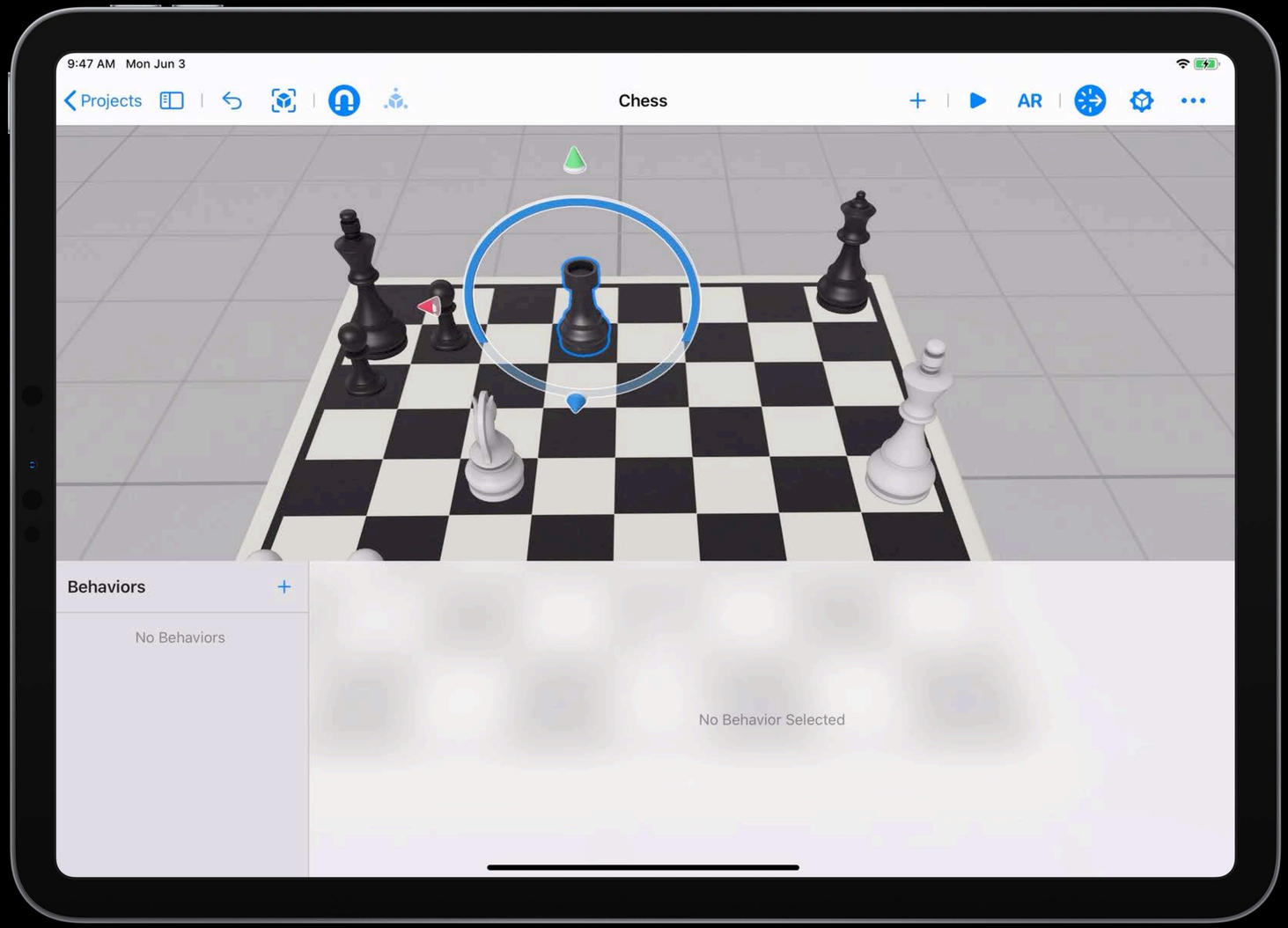
Simple Interactions



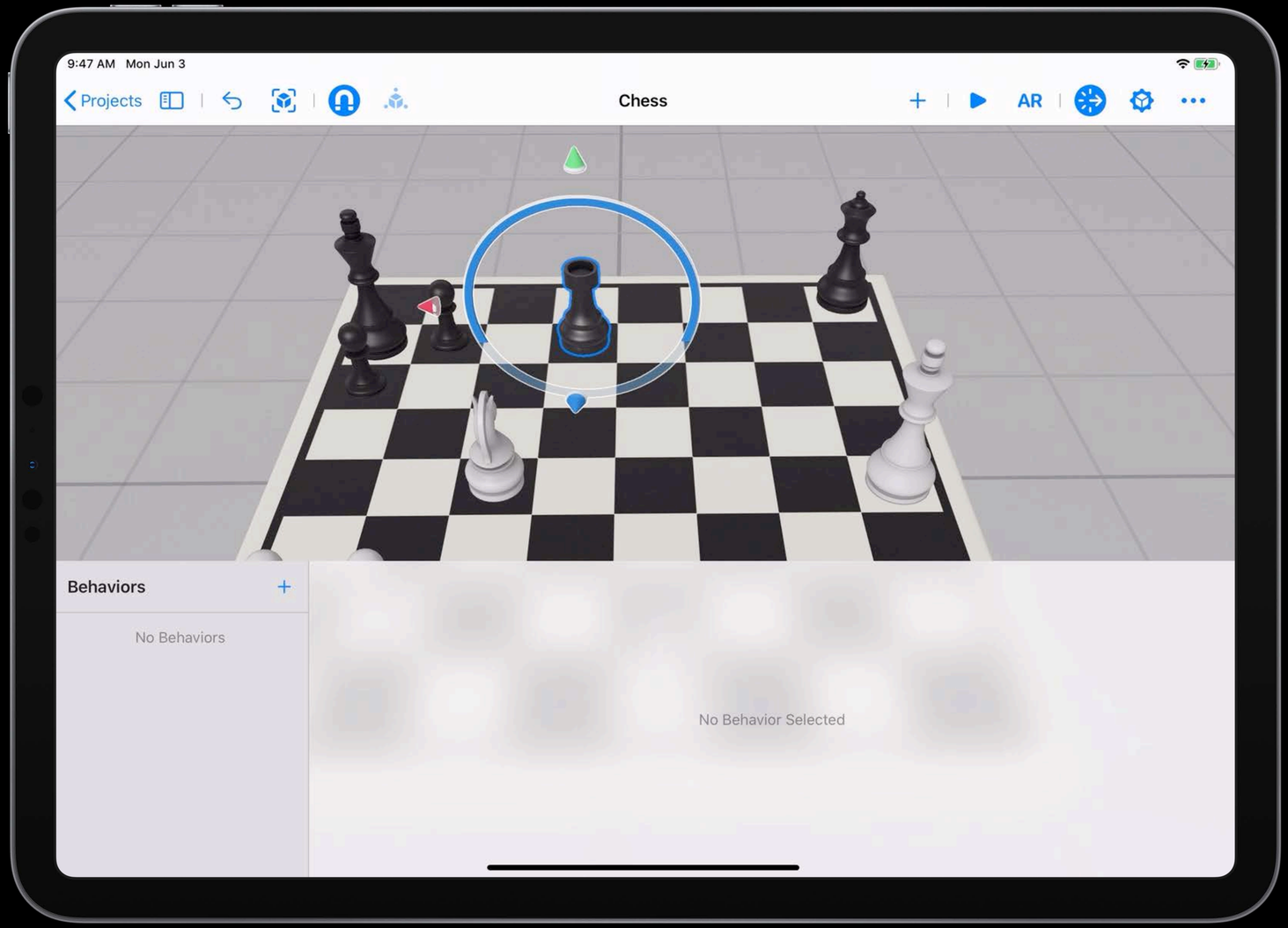
Simple Interactions



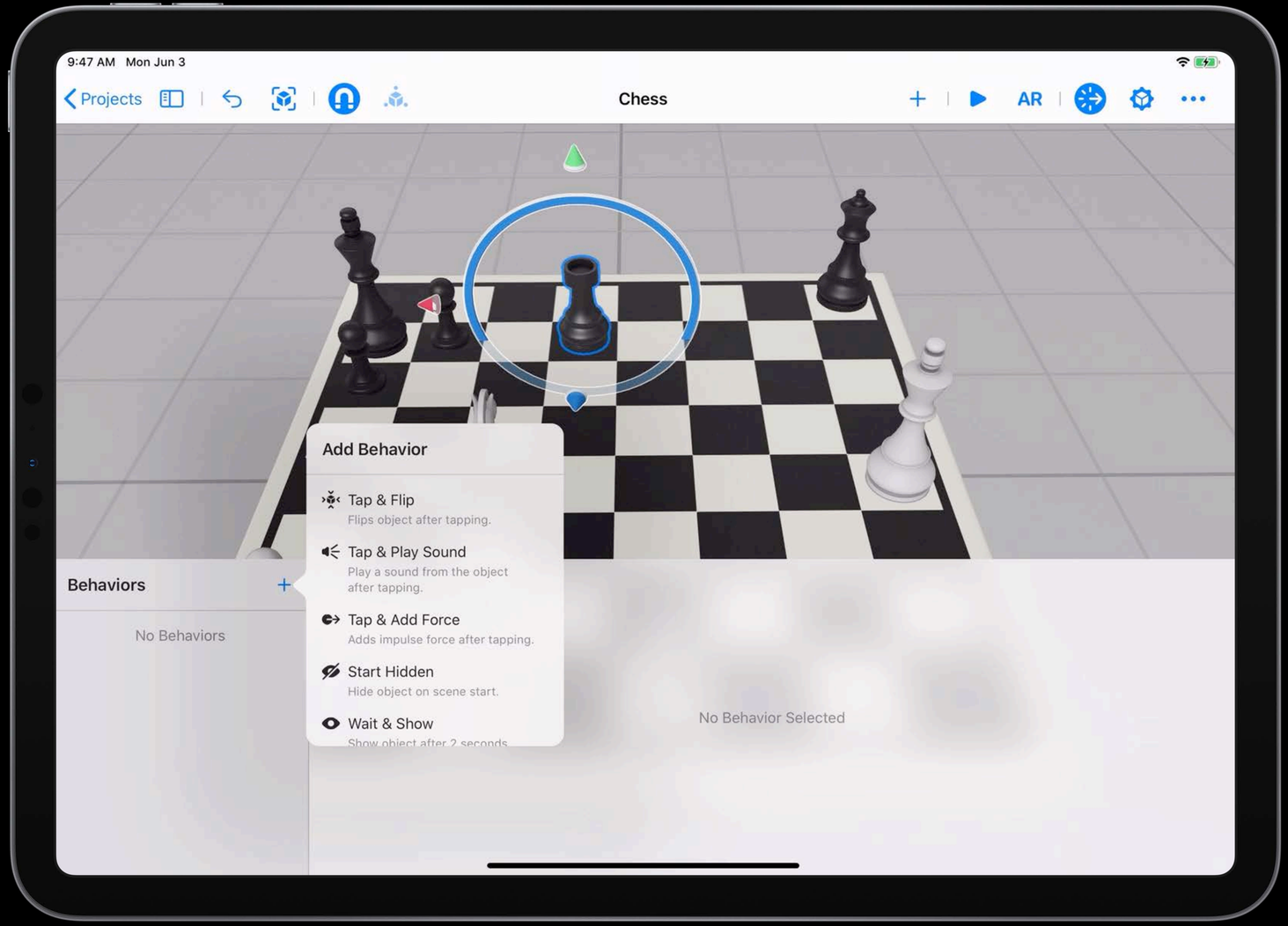
Simple Interactions



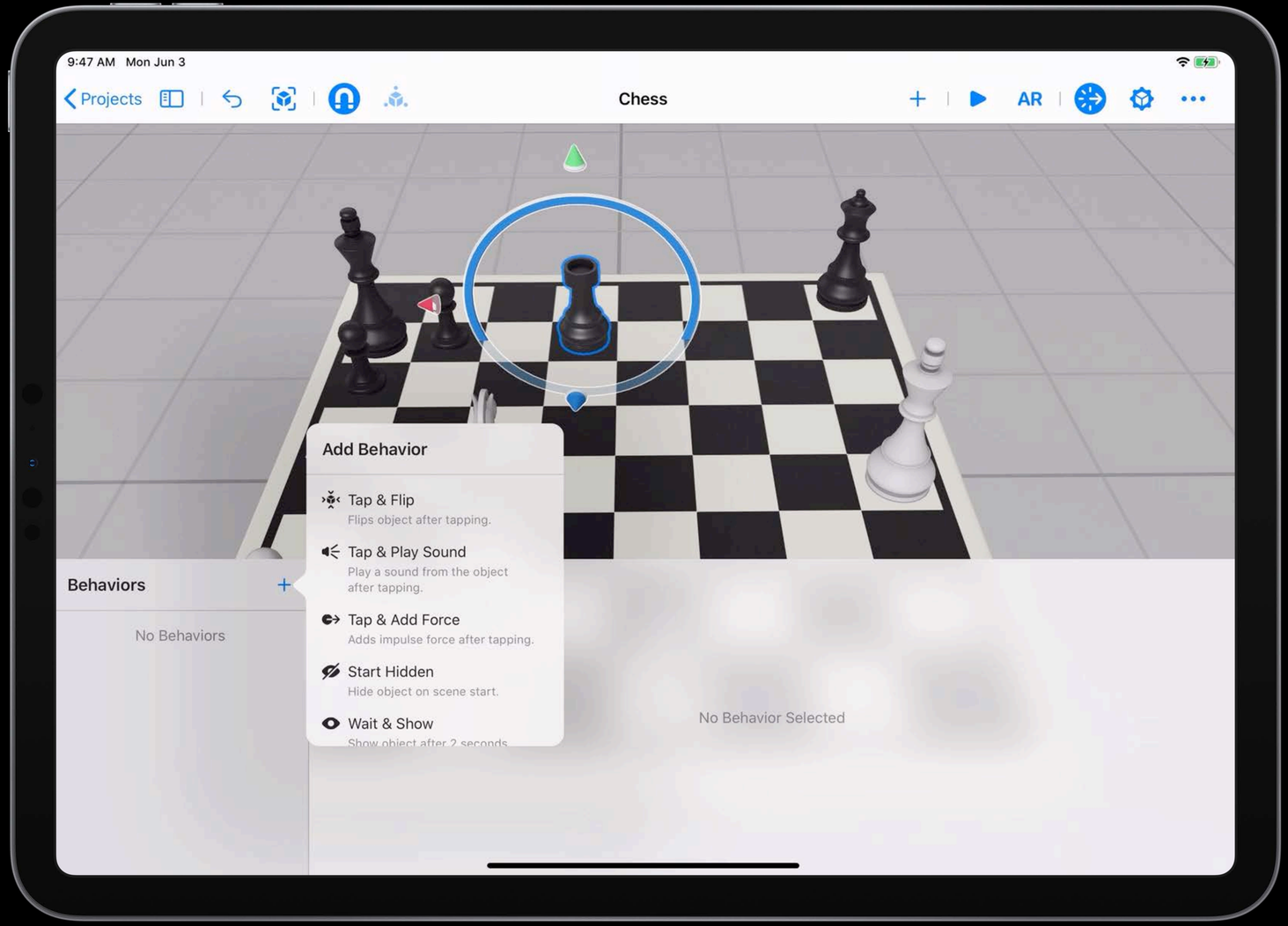
Simple Interactions



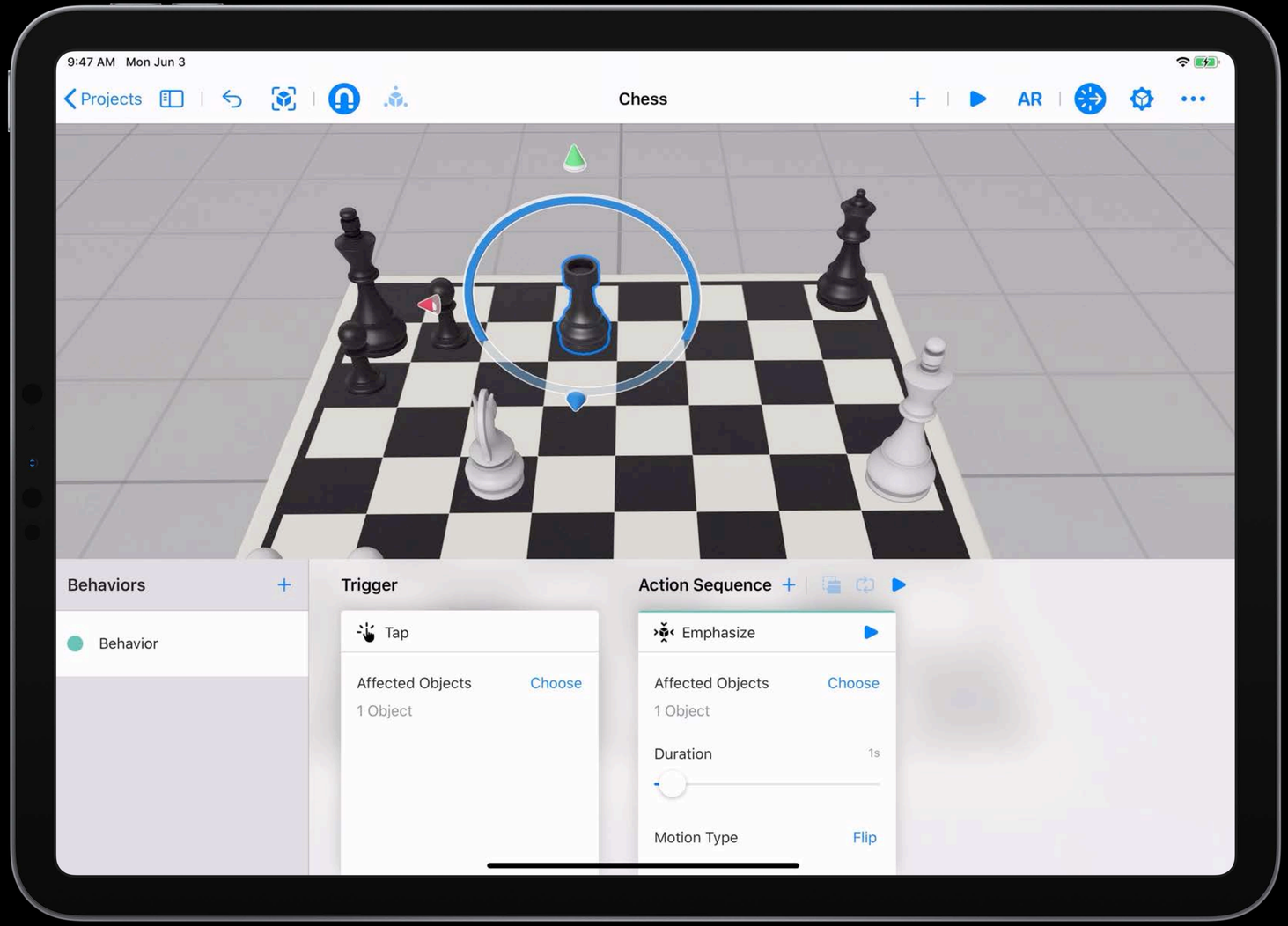
Simple Interactions



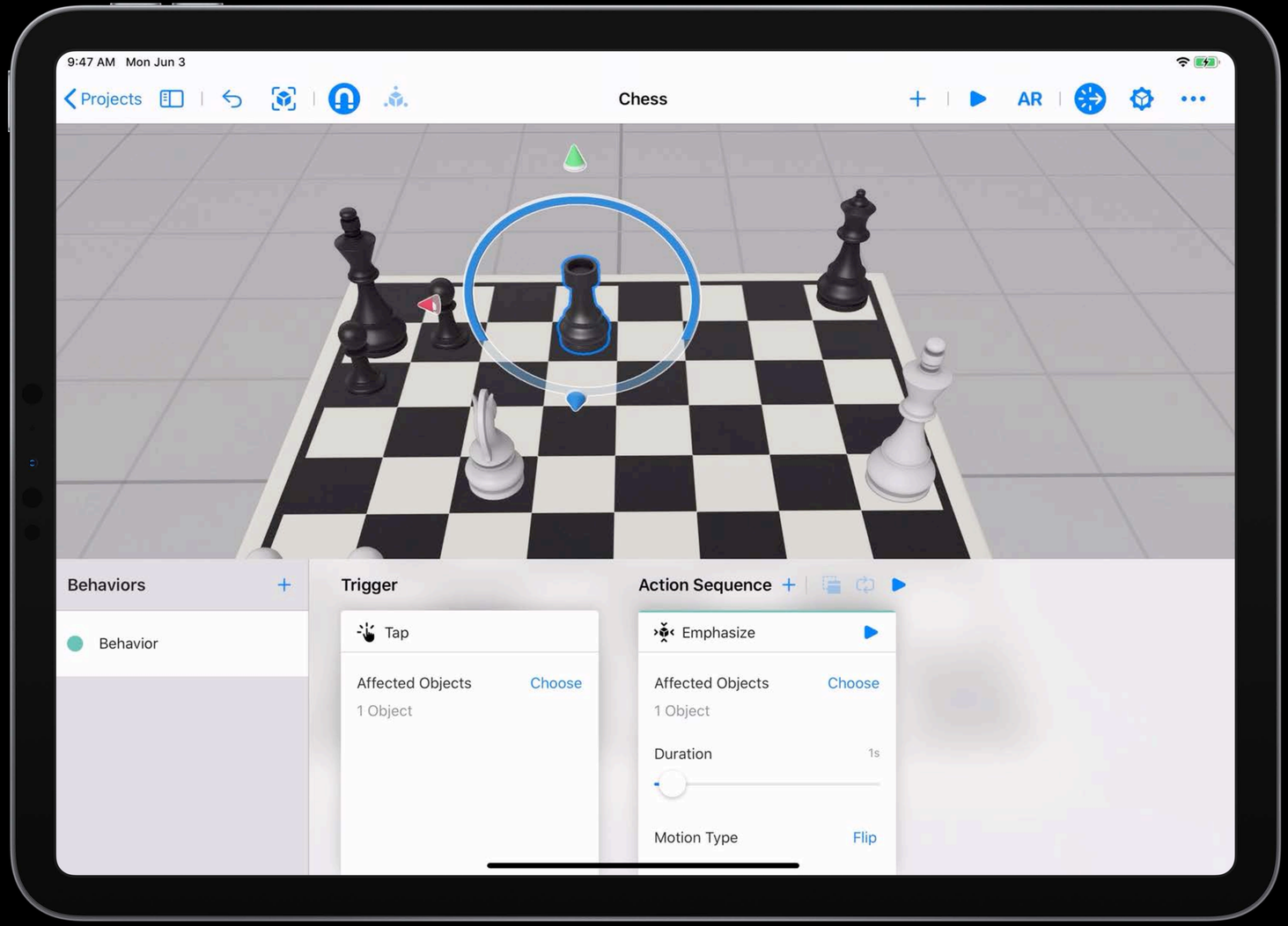
Simple Interactions



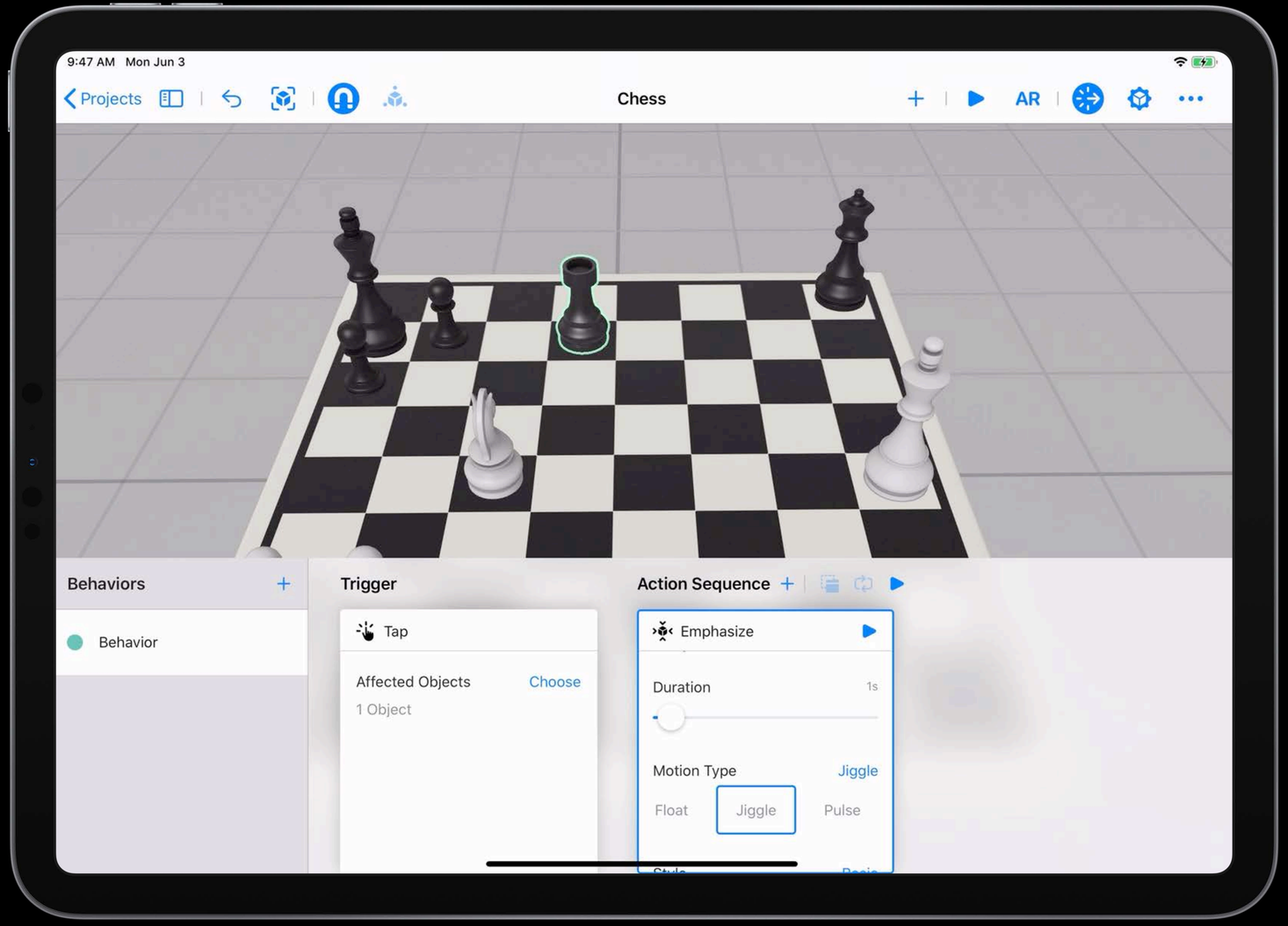
Simple Interactions



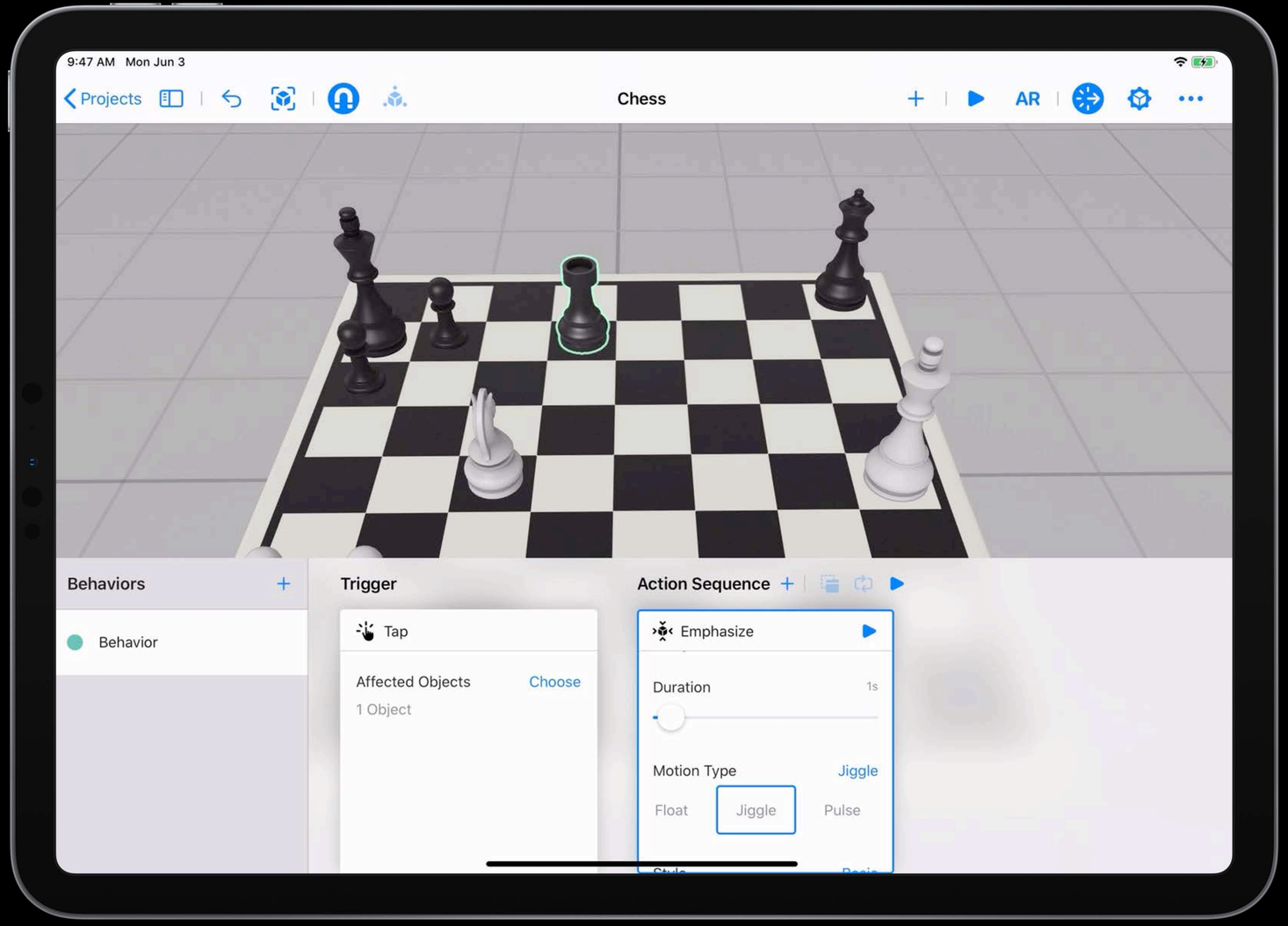
Simple Interactions



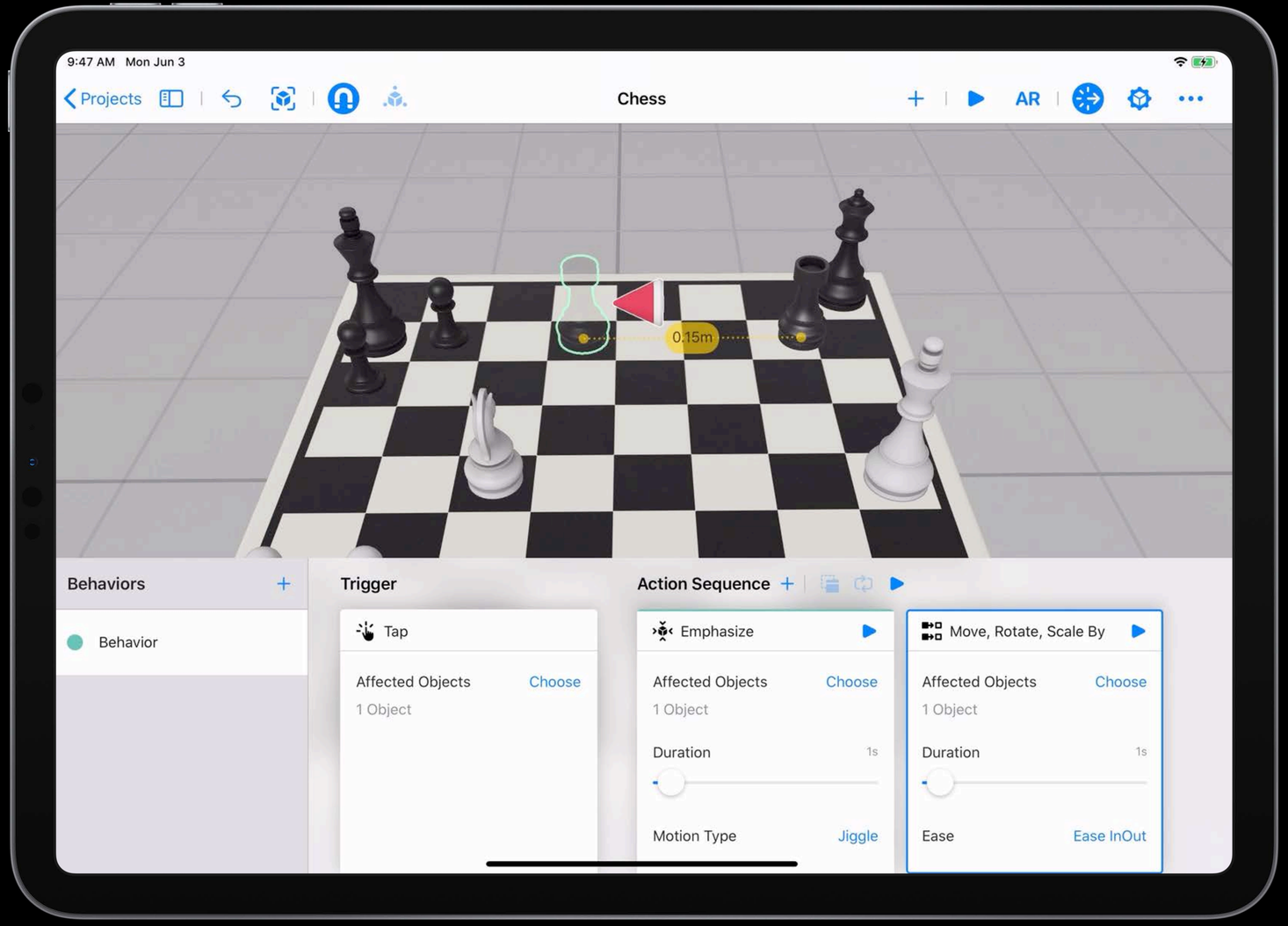
Simple Interactions



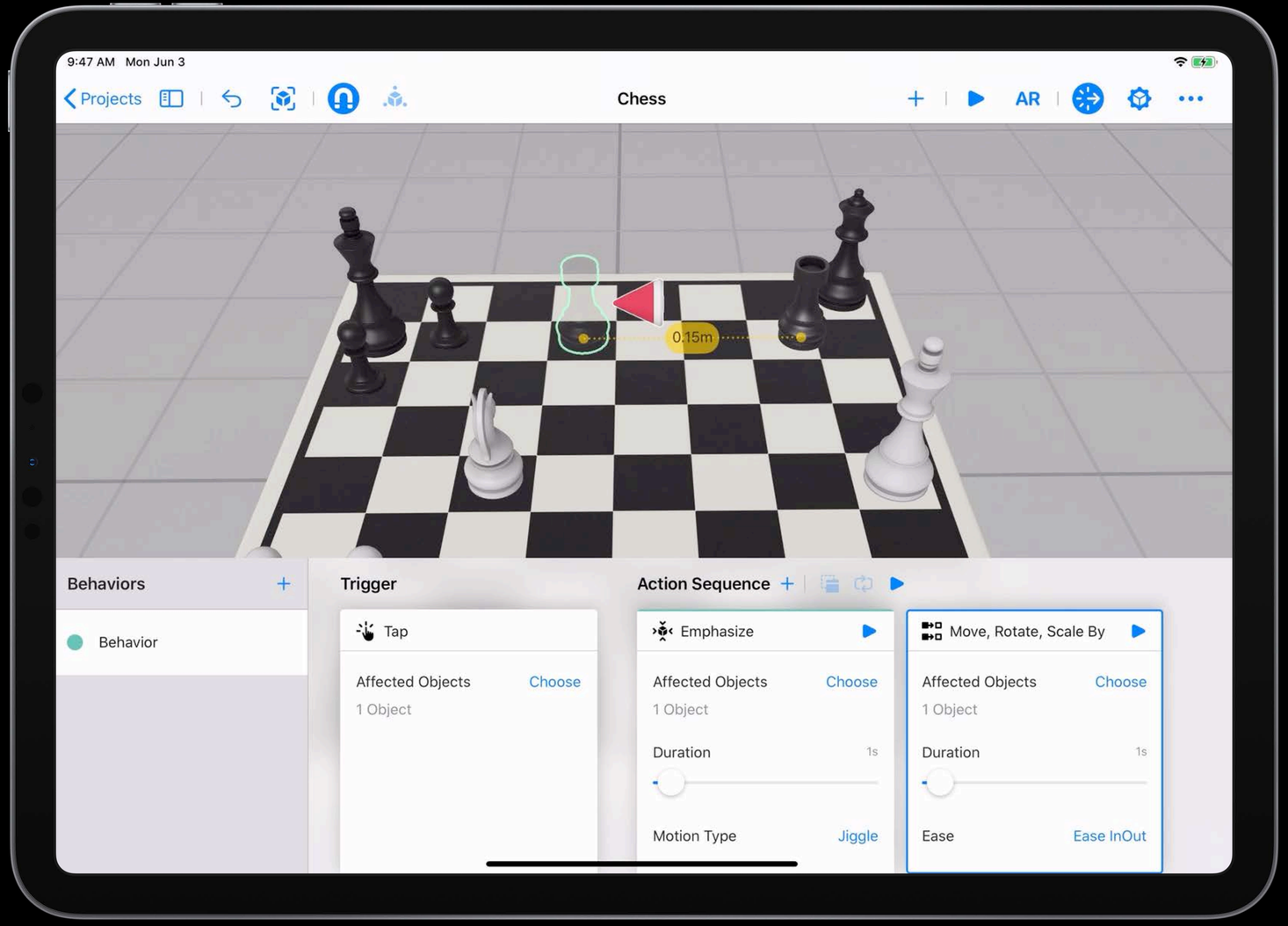
Simple Interactions



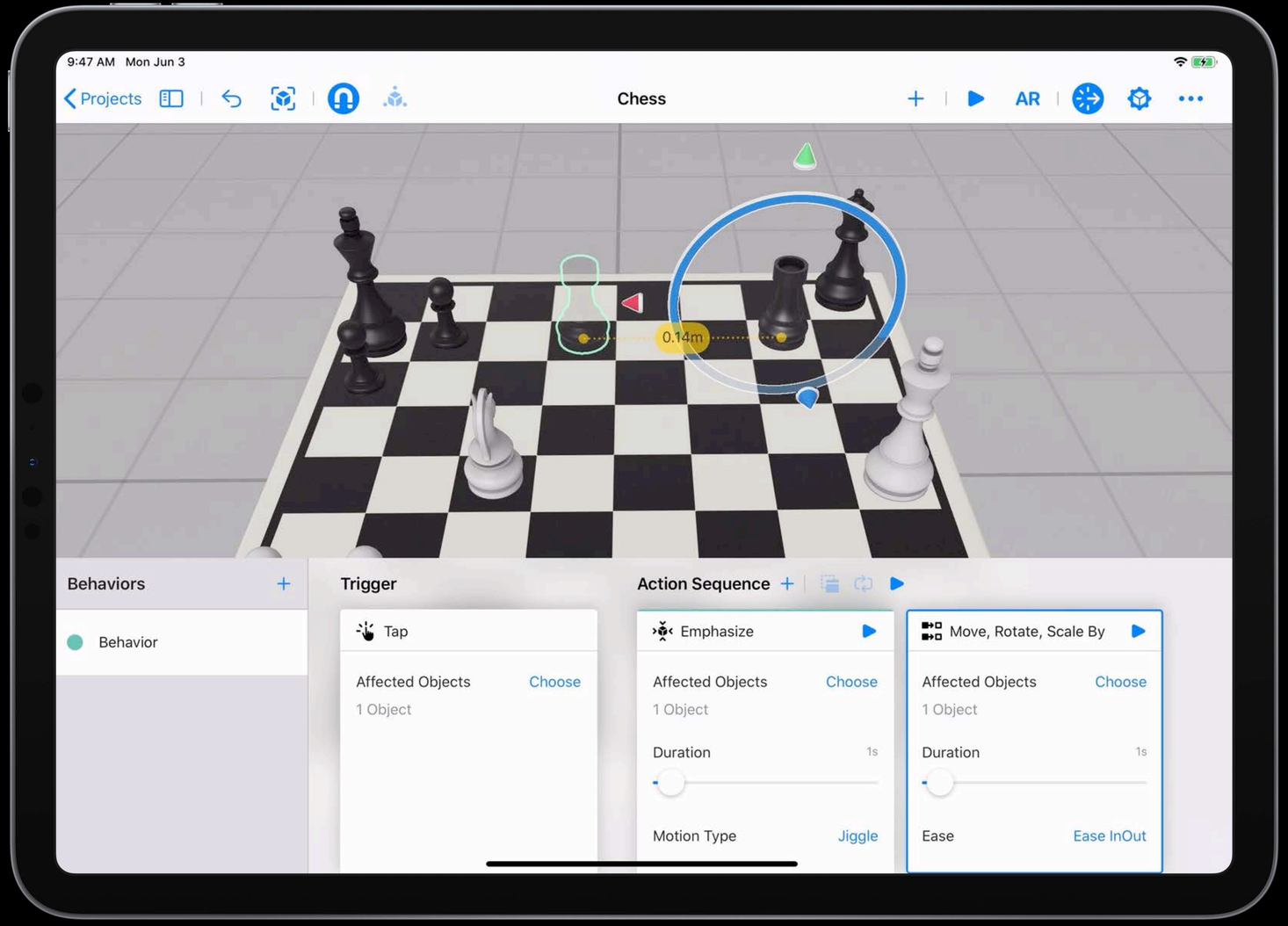
Simple Interactions



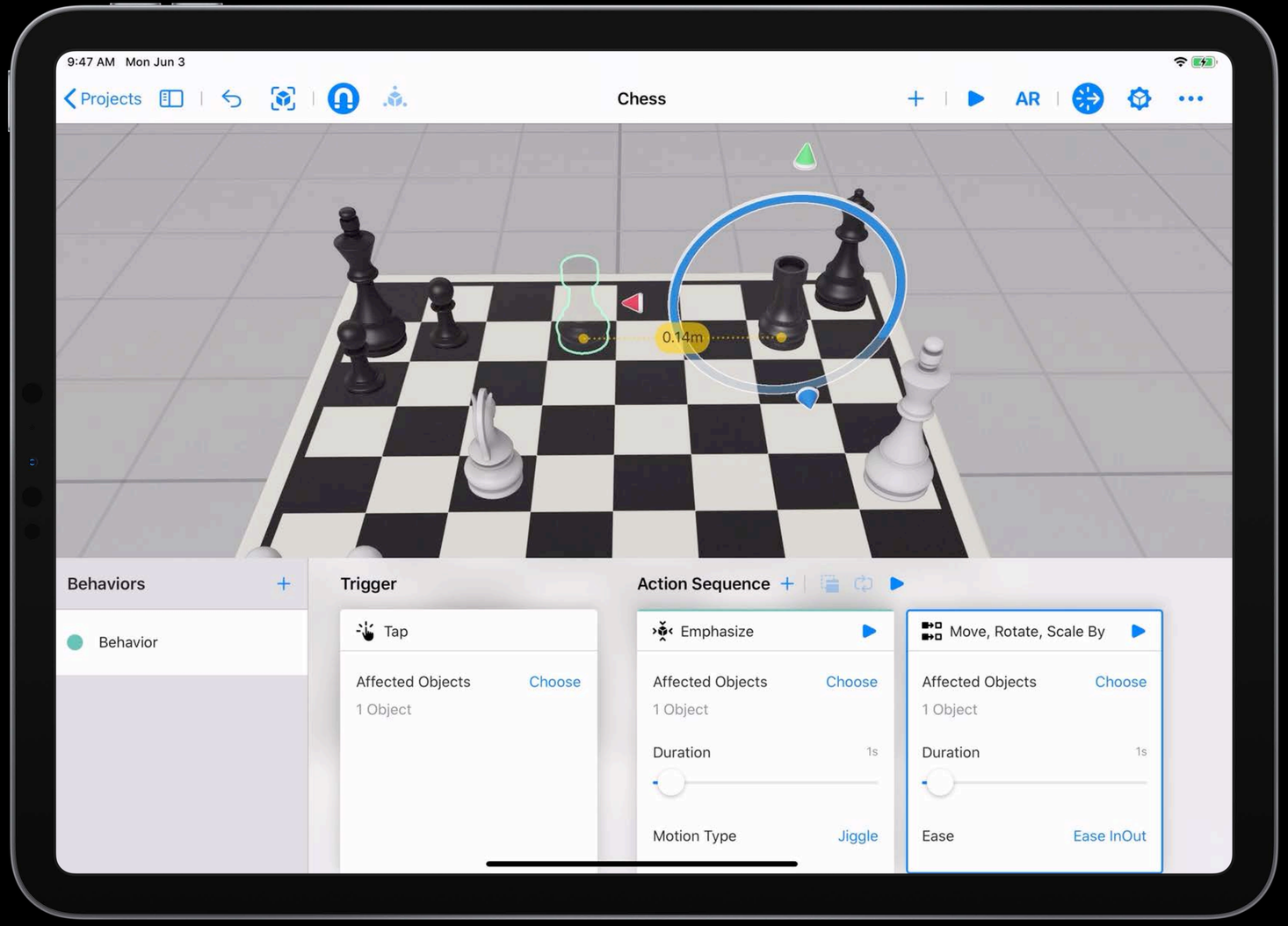
Simple Interactions



Simple Interactions



Simple Interactions



Simple Interactions

Simple Interactions

Reality Composer in Xcode

Experience.rcproject

Xcode

Experience.app

Reality Composer in Xcode

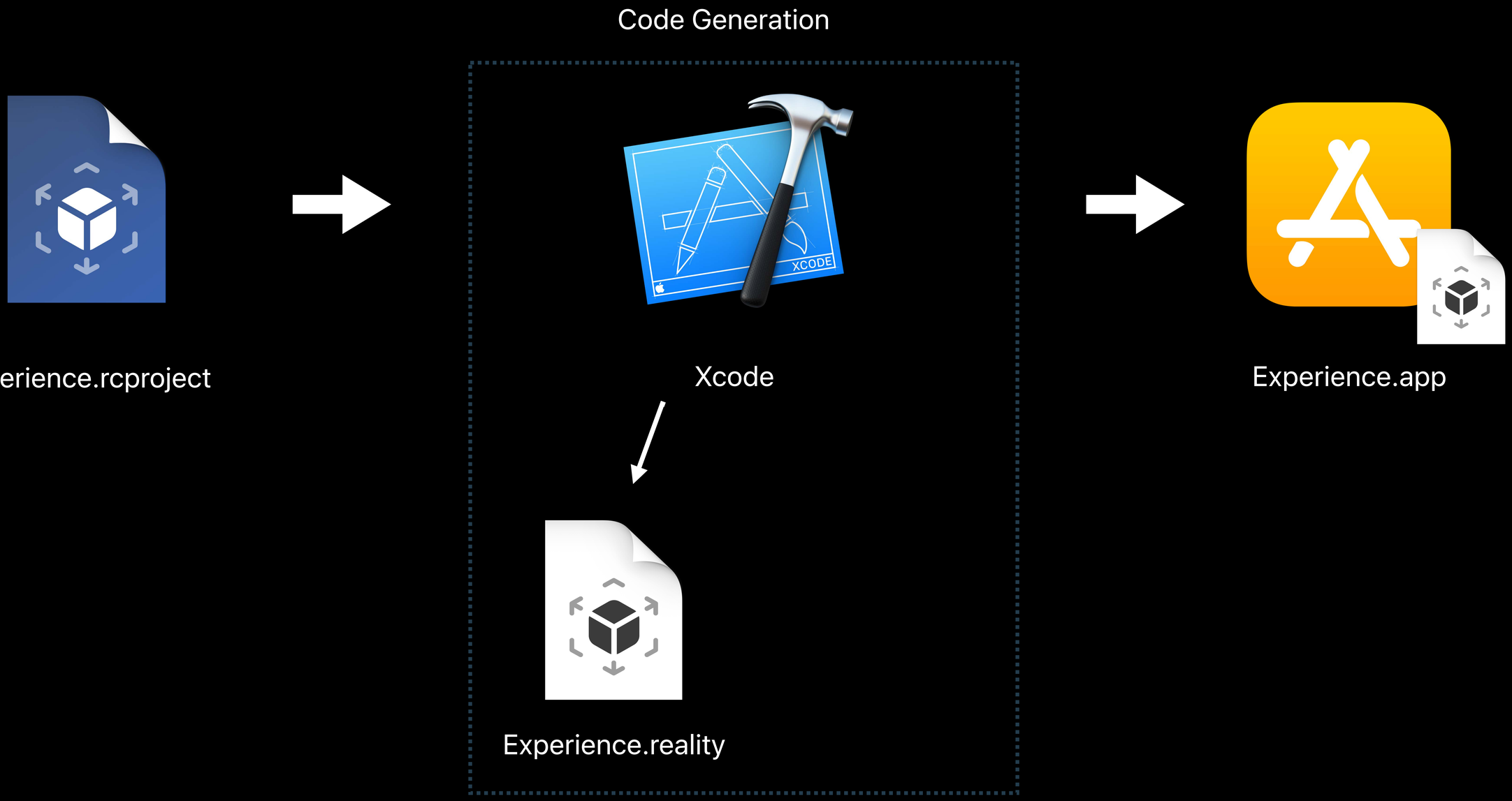
Code Generation

Experience.rcproject

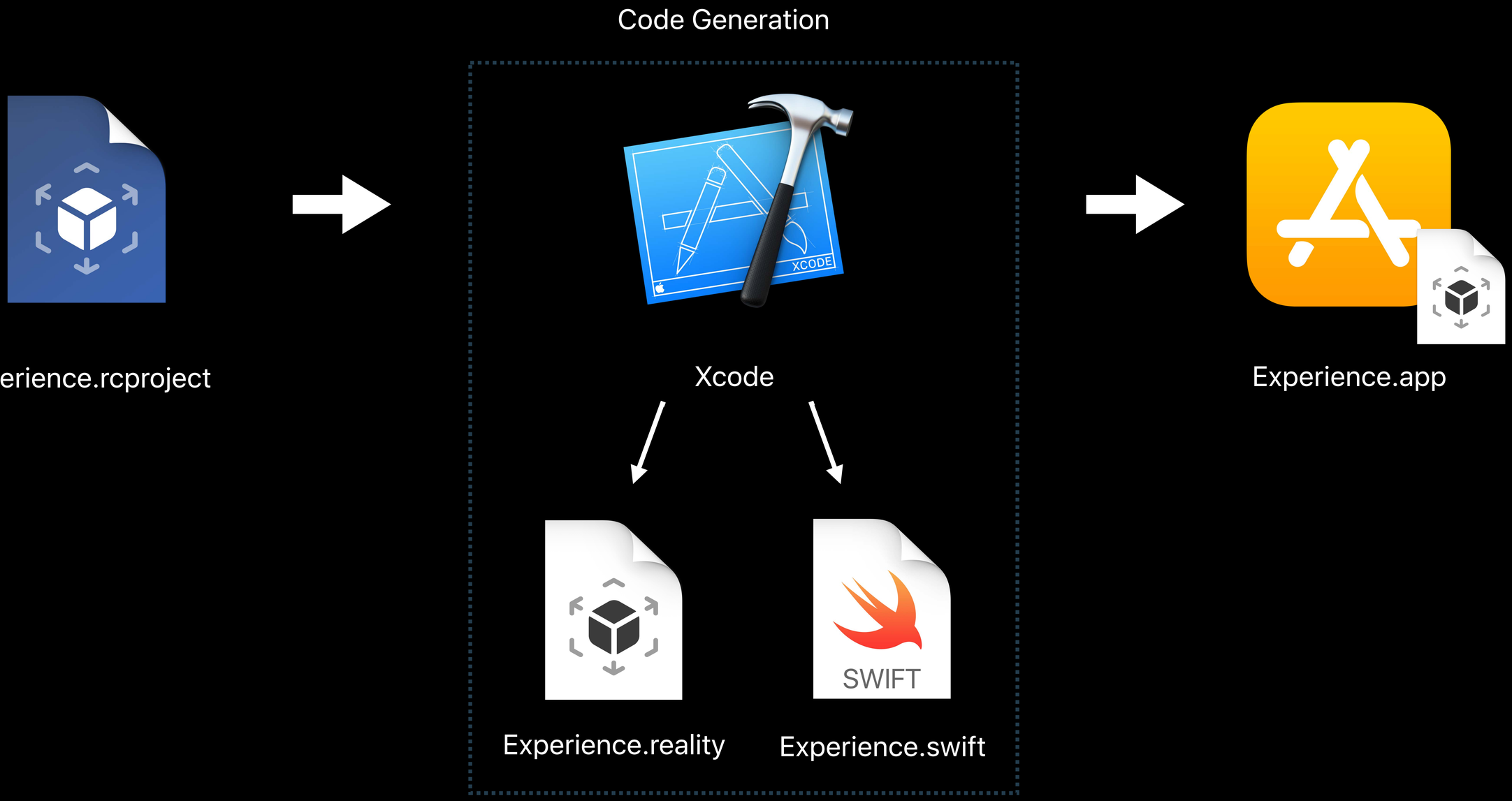
Xcode

Experience.app

Reality Composer in Xcode



Reality Composer in Xcode

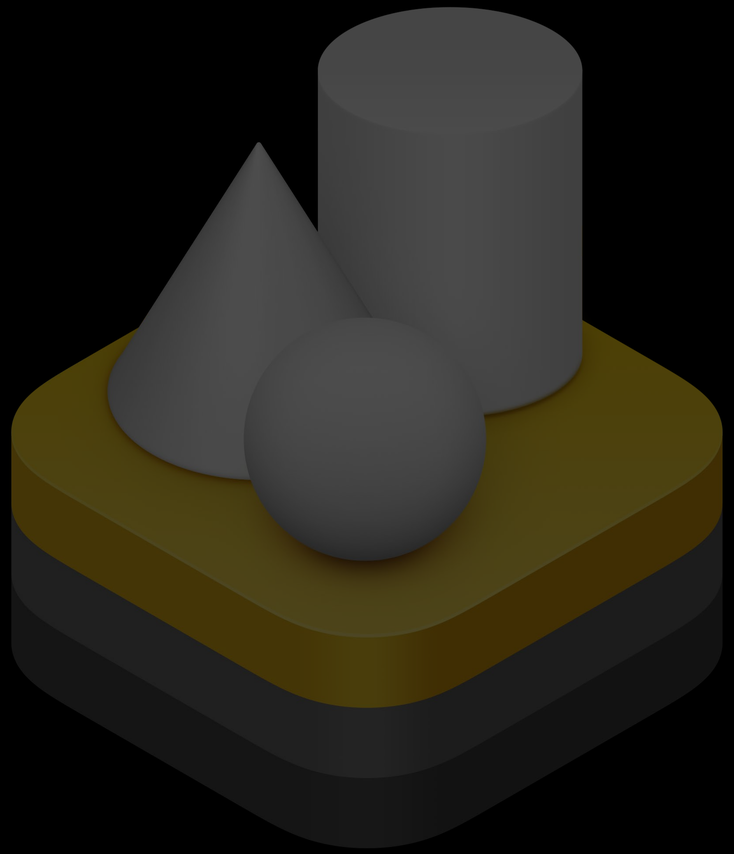


Code Generation

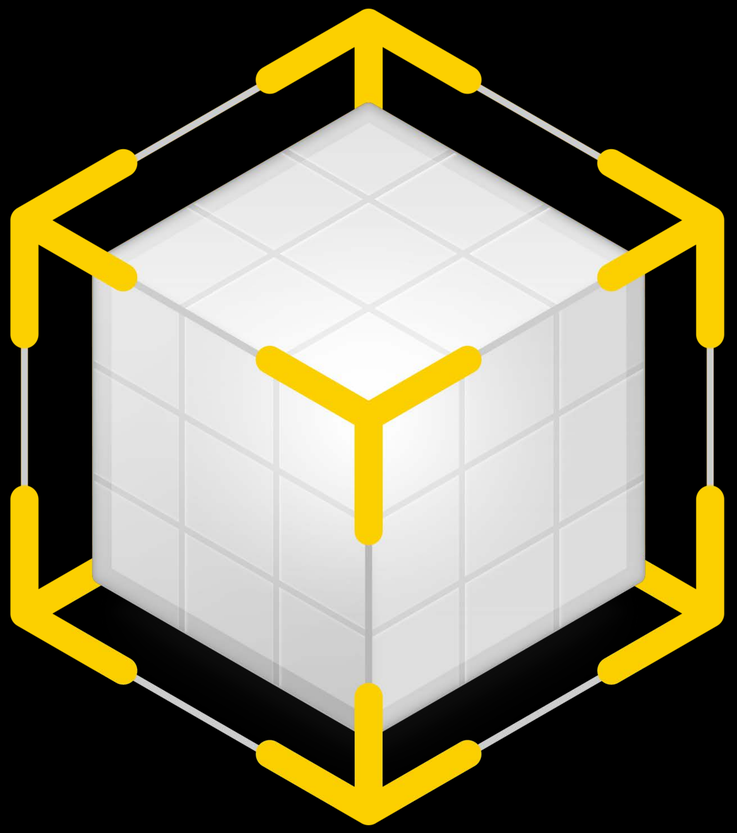
Generated by Xcode

Strongly-typed access

Customize behaviors

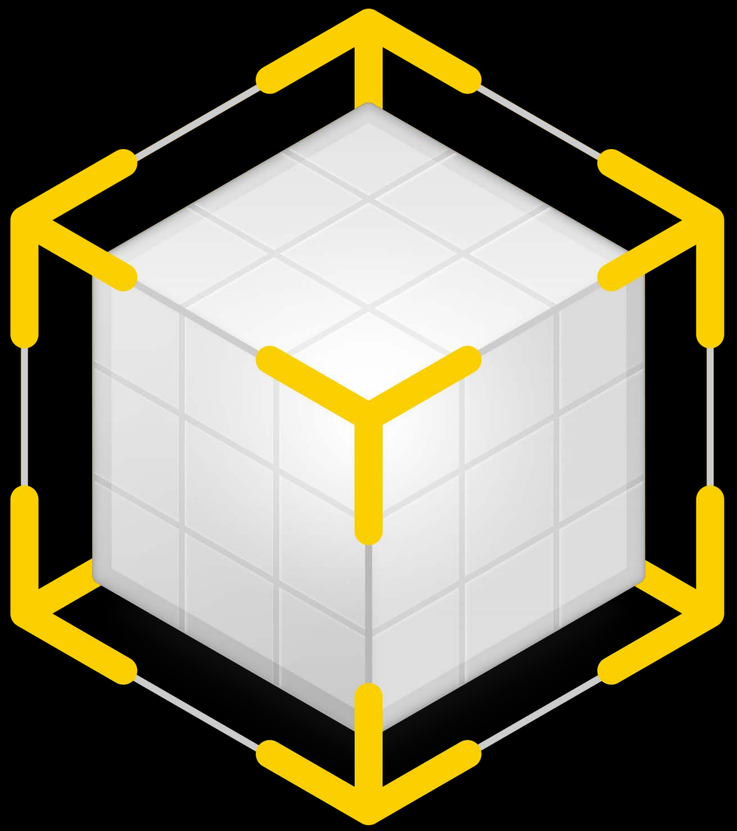


RealityKit

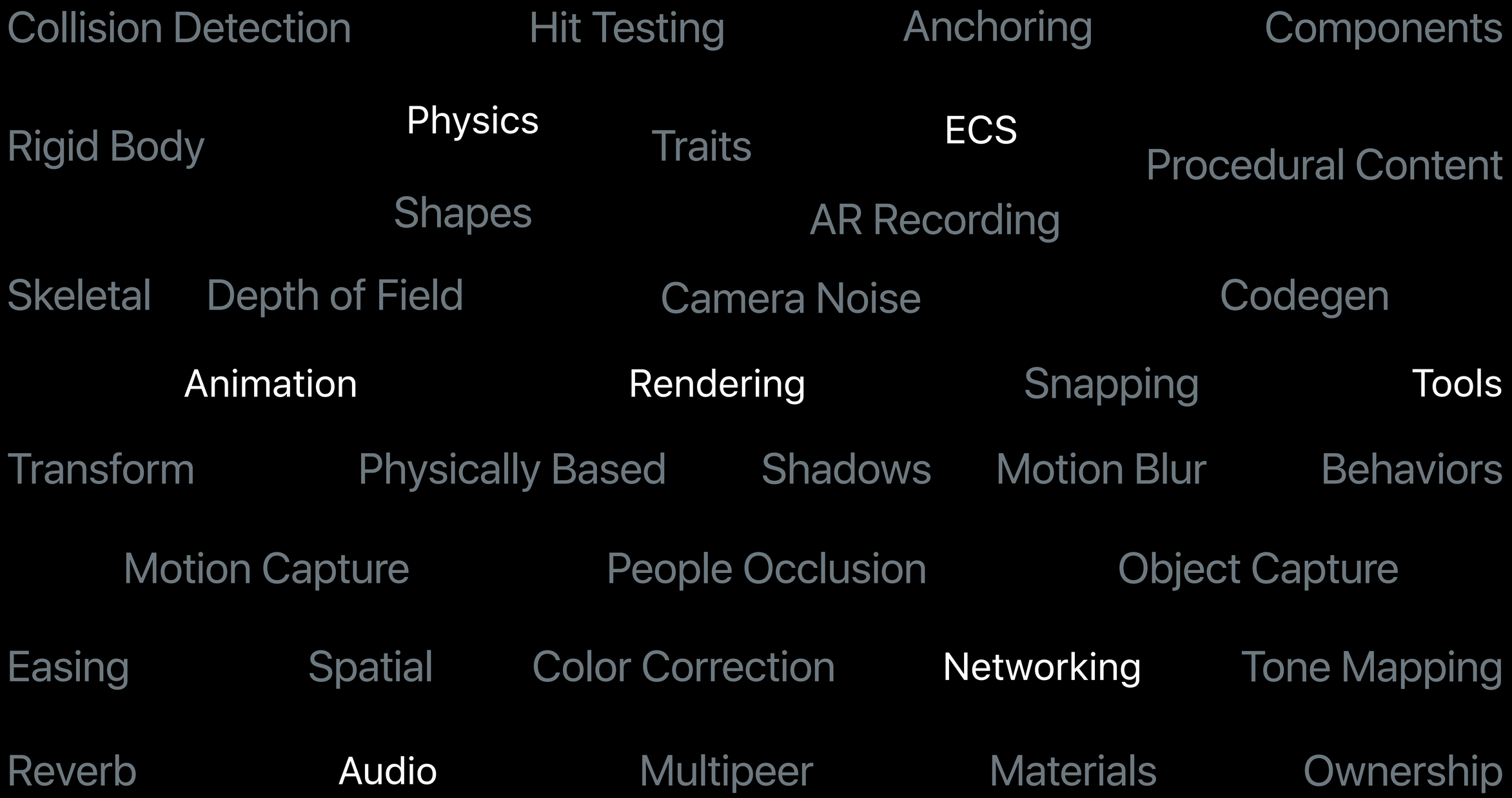


Reality Composer

RealityKit



Reality Composer


Collision Detection
Hit Testing
Anchoring
Components
Physics
Rigid Body
Traits
ECS
Procedural Content
Shapes
AR Recording
Skeletal
Depth of Field
Camera Noise
Codegen
Animation
Rendering
Snapping
Tools
Transform
Physically Based
Shadows
Motion Blur
Behaviors
Motion Capture
People Occlusion
Object Capture
Easing
Spatial
Color Correction
Networking
Tone Mapping
Reverb
Audio
Multipeer
Materials
Ownership

More Information

developer.apple.com/wwdc19/603

Building Apps with RealityKit	Wednesday, 10:00
Building AR Experiences with Reality Composer	Thursday, 11:00
RealityKit and Reality Composer Lab	Wednesday, 12:00


WWDC19