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

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

Mesh
Physics
Animation
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  Anchor  Scene  Entity
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
Entity

Building block of every AR object
Establishes scene structure
Provides transform hierarchy
AR Anchoring
AR Anchoring

Horizontal

Vertical
AR Anchoring

- Horizontal
- Vertical

Anchor

Anchor
AR Anchoring
AR Anchoring
AR Anchoring
AR Anchoring

Horizontal

Vertical
ARView

Scene

Anchor

Anchor

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

- Ball
- Lamp
- Camera
Entities and Components

- Ball
- Lamp
- Camera

Anchoring
Entities and Components

- Ball
  - Anchoring
  - Model

- Lamp
  - Anchoring
  - Model

- Camera
  - Anchoring
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)

/// 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)

/// 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)

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

entity.components[SpotLightComponent.self] = SpotLightComponent(color: .white)

entity.components[CollisionComponent.self] = nil

disable.addChild(childEntity)

entity.position = [1.0, 0.0, 0.5]

entity.setPosition([0.5, 0.2, 1.5], relativeTo: nil)
/// Create new entity
let entity = Entity()

/// Add/modify component
defaultComponents[SpotLightComponent.self] = SpotLightComponent(color: .white)

/// Remove component
defaultComponents[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)

/// 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)
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

```swift
let tableAnchor = AnchorEntity(plane: .horizontal,
                               classification: .table,
                               minimumBounds: [0.5, 0.5])
```

Add table anchor to scene

```swift
scene.addAnchor(tableAnchor)
```

Create an anchor to a reference image

```swift
let imageAnchor = AnchorEntity(.image(group: "Posters", name: "WorldsFair"))
```

Add image anchor to scene

```swift
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)
Mesh Resource

Provides geometry of scene
Shared across entities
Built-in primitives
Primitive Meshes

Box

Sphere

Plane

Text

Hello world.
ModelEntity

Mesh Resource

Material

Material

Material
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

- `baseColor` 0.2
- `roughness` 0.2
- `metallic` 0.0
SimpleMaterial

- baseColor: 0.2
- roughness: 0.0
- metallic: 0.0
SimpleMaterial

- baseColor
- roughness: 0.2
- metallic: 0.0
SimpleMaterial

baseColor  roughness  metallic
0.2        0.0        1.00

SimpleMaterial

- baseColor
- roughness: 0.2
- metallic: 1.0
SimpleMaterial

baseColor

roughness: 0.2

metallic: 1.0
SimpleMaterial

- baseColor
- roughness: 0.8
- metallic: 1.0
SimpleMaterial

- baseColor: 0.8
- roughness: 0.2
- metallic: 1.0
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

Mesh Resource

Material

Material

Material
Create anchor entity for attaching content

```swift
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)
```

Generate a box mesh

```swift
let box = MeshResource.generateBox(size: 1.0, cornerRadius: 0.1)
```

Create a simple metallic gray material

```swift
let metal = SimpleMaterial(color: .gray, isMetallic: true)
```

Create a model entity from mesh and material

```swift
let model = try ModelEntity(mesh: box, materials: [metal])
anchor.addChild(model)
```
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)

let box = MeshResource.generateBox(size: 1.0, cornerRadius: 0.1)

let metal = SimpleMaterial(color: .gray, isMetallic: true)

let model = try ModelEntity(mesh: box, materials: [metal])
anchor.addChild(model)
let anchor = AnchorEntity(plane: .horizontal)
scene.addAnchor(tableAnchor)

let box = MeshResource.generateBox(size: 1.0, cornerRadius: 0.1)

let metal = SimpleMaterial(color: .gray, isMetallic: true)

let model = try ModelEntity(mesh: box, materials: [metal])
anchor.addChild(model)
Animation

Supports skeletal and transform

Load from ‘usdz’ or Reality File
let controller = entity.playAnimation(named: "dance")

controller.pause()

if controller.isPaused {
    controller.resume()
}

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

/// Pause animation
controller.pause()

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

/// Stop animation
controller.stop()
let controller = entity.playAnimation(named: "dance")

controller.pause()

if controller.isPaused {
    controller.resume()
}

controller.stop()
let controller = entity.playAnimation(named: "dance")

controller.pause()

if controller.isPaused {
    controller.resume()
}

controller.stop()
let controller = entity.playAnimation(named: "dance")

controller.pause()

if controller.isPaused {
    controller.resume()
}

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
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
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

Experience.rcproject → Code Generation → Xcode → Experience.app
Reality Composer in Xcode

- Experience.rcproject
- Xcode
- Experience.swift
- Experience.app
Code Generation

Generated by Xcode
Strongly-typed access
Customize behaviors
## More Information

developer.apple.com/wwdc19/603

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building Apps with RealityKit</td>
<td>Wednesday, 10:00</td>
</tr>
<tr>
<td>Building AR Experiences with Reality Composer</td>
<td>Thursday, 11:00</td>
</tr>
<tr>
<td>RealityKit and Reality Composer Lab</td>
<td>Wednesday, 12:00</td>
</tr>
</tbody>
</table>