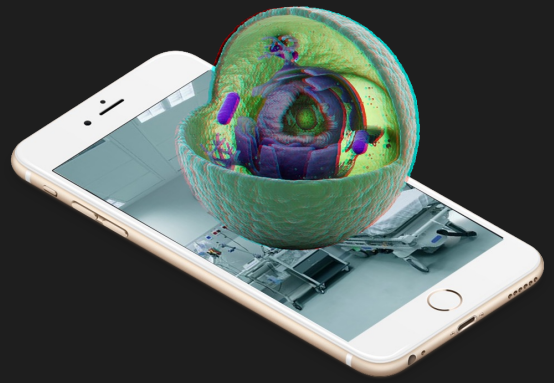


RealityKit

iPad & iPhone

KEEPCODING
Tech School

Introducción

- ARKit de Apple se construyó sobre la base de SceneKit, que es un marco de Gráficos 3D para juegos móviles. Se usaba en una primera versión el motor de renderizado de SceneKit.

- RealityKit se introdujo después de ARKit como una forma de simplificar el trabajo de creaciones de aplicaciones de Realidad aumentada con muchas funciones como Gestos, colisiones e iluminación de objetos.

Introducción

- RealityKit Fue desarrollado desde cero para simplificar y acelerar el desarrollo de realidad aumentada, renderizado, efectos de cámara, animaciones, físicas etc.

https://developer.apple.com/documentation/realitykit/

SceneKit – iOS 8

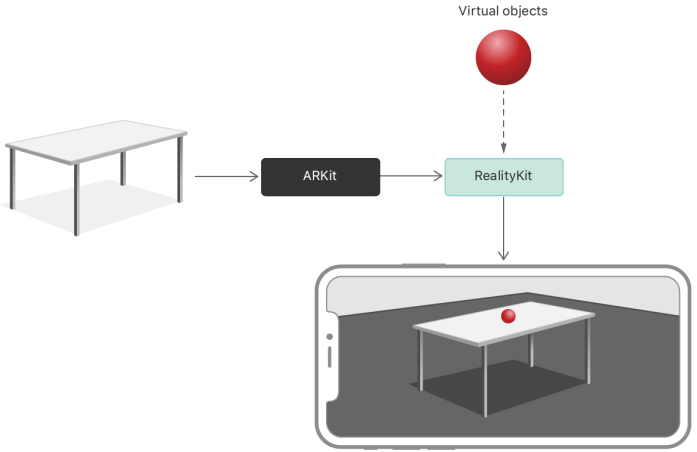
- Se desarrollo originalmente para juegos móviles
- Renderiza los datos de ARKit para representar objetos de forma realista
- SceneKit proporciona la forma de gestionar dichos objetos

ARKit – iOS 11

- ARKit captura datos del dispositivo para representar los datos de forma realista
- Combina el seguimiento de movimientos del dispositivo, capturas de la escena de la cámara y el procesado avanzado de la escena.

RealityKit – iOS 13

- Se crea para la simplificación de creación de aplicaciones de realidad aumentada
- Proporciona un montón de funciones de alto nivel como la gestión de gestos, rotaciones, físicas etc. que con ARKit requería cálculos y renderizados de forma manual.



RealityKit – Requerimientos

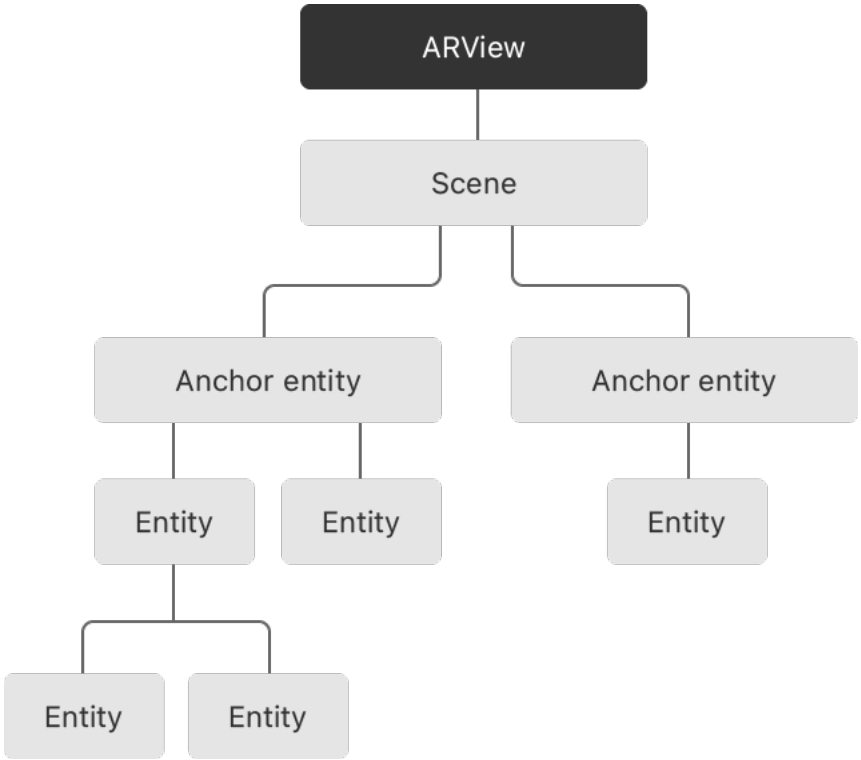
- Se necesita un iphone o iPad para desarrollar y debe estar conectado a Xcode en el momento de compilar
- Se debe probar contra el dispositivo físico no simulador
- Procesador A12 mínimo
- Si tienes un iPhone Pro o IPad Pro será más rápido y preciso porque incorporan Lidar.

¿Qué es Lidar?

- Esta tecnología detecta la luz y la distancia que la separa de los objetos. Conjunto de laser que se usan para determinar la distancia que tenemos con los objetos
- Lidar = Light detection and ranging
- Apple lo usa para mejorar las fotos y la realidad aumentada
- Fue creado en los años 60 por Hughes Aircraft Company y años después la NASA lo empezó a usar en sus proyectos.
- Los robots aspiradora lo usan para hacer un mapa 3D de la casa y aspirar de forma más inteligente

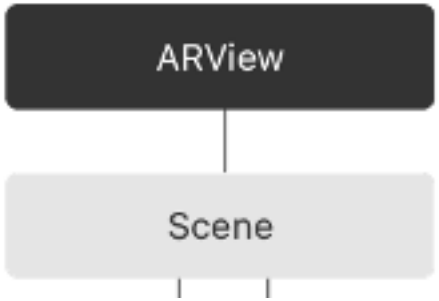
RealityKit

Objetos al programar RealityKit



ARView

- Se usa una instancia de ARView para mostrar y renderizar objetos 3D al usuario
- Normalmente se añade una sola vista a la app
- Una vista tiene una única instancia de Escena (Scene)



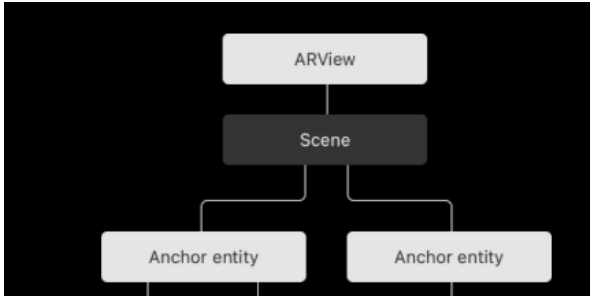

```
struct ARViewContainer: UIViewRepresentable {

    func makeUIView(context: Context) -> ARView {

        //ArView object
        let arView = ARView(frame: .zero)
```

Scene (escena)

- Se obtiene la escena desde una instancia de ARView
- Para añadir objetos 3D primero hay que añadirle a la escena un ancla (Anchor)
- Los Anchor le indican a RealityKit como fijar el contenido en el mundo real, como superficies, imágenes, caras etc.
- Un Scene puede tener muchos Anchor

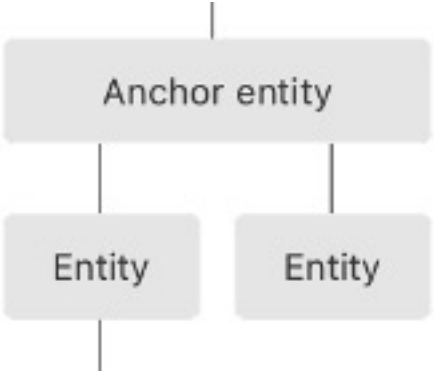



```
//Creamos un ancla
let anchor = AnchorEntity(plane: .horizontal) //anchamos a la horizontal

//Añadimos el ancla a la scena
arView.scene.anchors.append(anchor)
```

Anchor (ancla)

- Se crean para indicar a Reality Kit donde debe anclar en el mundo real los objetos que tiene dentro.
- Contine uno o N entidades (objetos 3D)
- Se le indica la posición en la Scene, rotación y escala de la entidad



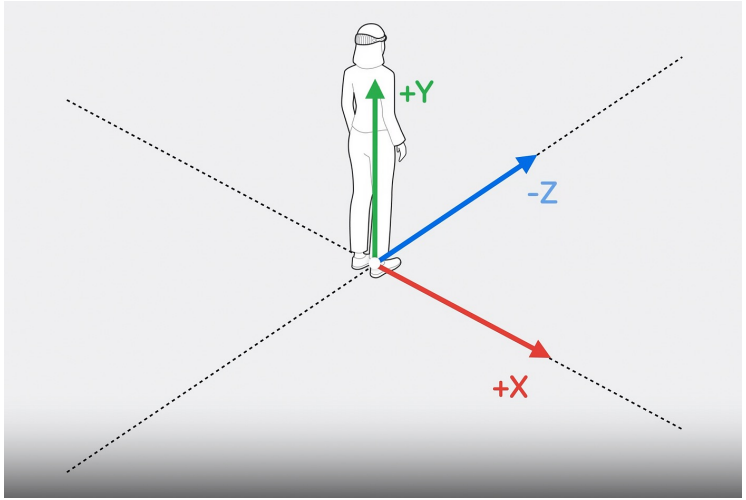

```swift
//ArView object
let arView = ARView(frame: .zero)

//Creamos un ancla
let anchor = AnchorEntity(plane: .horizontal) //anchamos a la horizontal

//Creamos una caja a mano (de 0,3 metros) (la entidad)
let material = SimpleMaterial(color: .blue, isMetallic: true) //Creamos el tipo de material de la caja
let box = ModelEntity(mesh: MeshResource.generateBox(size: 0.3), materials: [material])

//Añadimos la entidad al ancla
anchor.addChild(box)
//Añadimos el ancla a la scena
arView.scene.anchors.append(anchor)
```

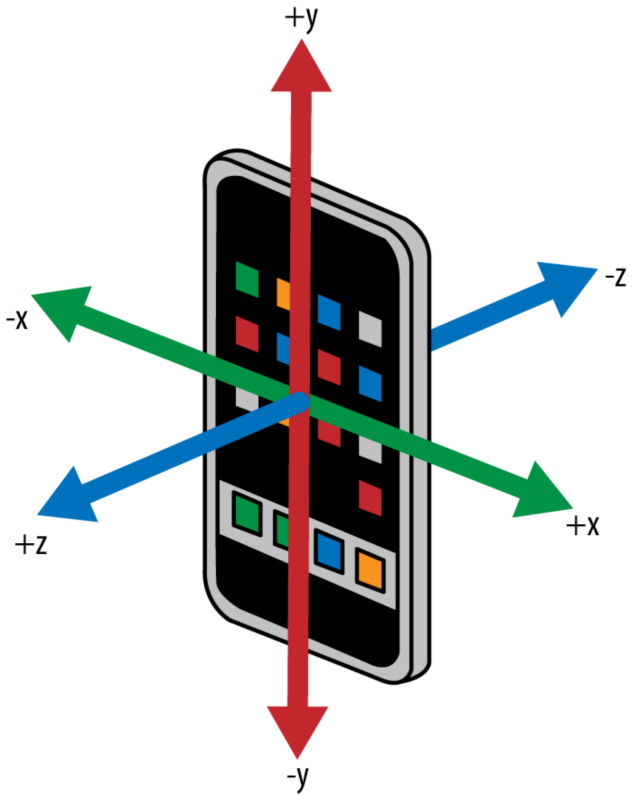
Planos 3D



- Posición Inicial es X, Y y Z = 0
- Cuando el dispositivo detecta el plano, lo marca como posición (0,0,0)
- Si a una entidad le cambiado la “position”:
  - X negativo = izquierda
  - X positivo = derecha
  - Y = 0 , posición detección del plano.
  - Y negativo = por debajo , bajamos objeto
  - Y positivo = por arriba, subimos objeto
  - Z positivo = hacia atrás (paso a atrás)
  - Z negativo = hacia adelante (paso adelante)

```
box.position = simd_make_float3(0, 0.5, 0) ///X, Y, Z
```

Planos 3D – ¡Vamos a jugar!



- Detección del plano en 0,0,0 (plano horizontal)
- Cambiamos posición objeto:
  - X = 0.5 (en metros)
  - Y = 0.5
  - Z = -0,5
  - Z = 0
  - Y = 0
  - X = -0,5
  - Y = -0,5
  - X = 0
  - Y = 0

¿Por qué vamos a usar SwiftUI?

- Porque Apple ha apostado al 100% por SwiftUI
- VisionOS de las gafas Vision Pro se programa en SwiftUI
- ¿Se puede en UIKit? Por supuesto y habrá muchos proyectos en el mercado.

Let’s code!

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá

Datos de contacto

y