ARKit과의 통합 및 AR 세션 구성

김민송

ARKit과 RealityKit 연동의 필요성

- ARKit이 제공하는 핵심 기능
  - 카메리 트래킹, 평면 탐지, 환경 맵핑 등 -> 저수준 기능
- RealityKit이 제공하는 고수준 렌더링, 물리 엔진
- 두 프레임워크 통합 시 ARKit의 안정적인 트래킹 위에 RealityKit의 고수준 랜더링 및 물리 시뮬레이션을 쉽게 얹을 수 있음.

ARSession
개요

- 디바이스 카메라와 센서 이용
- 역할: 실시간 트래킹, 환경 데이터(조명, 깊이) 스트림
- 세션 lifeCycle: 실행 -> 업데이트 -> 일시정지 -> 종료
    - lifeCycle을 제대로 관리해야 성능 저하나 트래킹 손실 방지 가능
- Delegate callback을 통한 상태 관리

ARSession

실행 흐름

- 세션 초기화 ARSession()
- 구성 객체 설정 session.run(config)
- Delegate callback 수신 session(_:didUpdate:)
    - 프레임 데이터 수신
- 세션 옵션: 필요시 session.run(_:options:)로 anchor 재설정이나 기존 anchor 유지 등 동작 제어 (일시정지 / 재시작)

ARConfiguration

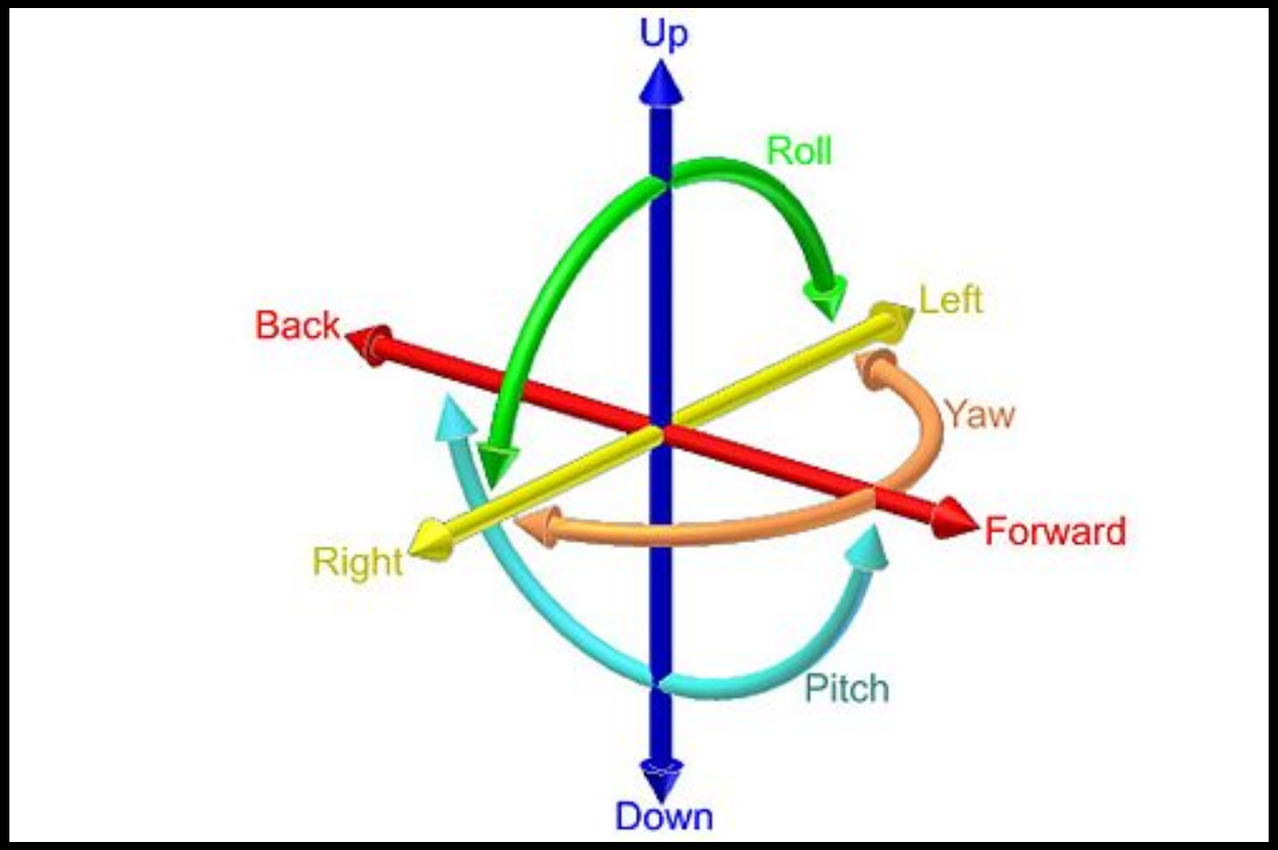
- ARWorldTrackingConfiguration

```
let config = ARWorldTrackingConfiguration()
```

  - 6DoF(world tracking)
    - 평면 감지, 이미지 인식, 환경 텍스쳐링
- ARImageTrackingConfiguration
  - 단일 / 다중 이미지 트래킹
- ARFaceTrackingConfiguration
  - 전면 카메라 얼굴 트래킹

ARConfiguration

6DoF



카메라 트래킹 모드

- worldAlignment
  - .gravity: 중력 방향에 뷰를 고정
  - .gravityAndHeading: 중력 + 나침반 방향까지 고려
  - .camera: 디바이스 카메라 기준 좌표계
- 실내 vs 실외 환경별 설정 전략 -> 트래킹 안정성

ARSession

실행 흐름

```swift
// ARSession 실행 (World Tracking, 평면 탐지, 환경 텍스처링 등)
private func startARSession() {
    let config = ARWorldTrackingConfiguration()
    config.planeDetection = [.horizontal, .vertical]
    config.environmentTexturing = .automatic
    //중력 + 나침반 방향까지 고려
    config.worldAlignment = .gravityAndHeading
    arView.session.run(config)
}
```

```swift
// 매 프레임마다 호출되는 업데이트 콜백
func session(_ session: ARSession, didUpdate frame: ARFrame) {
    if let estimate = frame.lightEstimate {
        // 조명 강도 반영
        directionalLight?.light.intensity = Float(estimate.ambientIntensity)
    }
}
```

ARView와 ARSession 연결

- RealityKit의 ARView 내부에 ARSession이 내장
- arView.session.delegate = self로 delegate 연결
- scene.anchors로 앵커 관리
- 랜더링 된 3D 콘텐츠가 자동으로 뷰에 반영됨.

```swift
// ARView 세팅 및 델리게이트 연결
private func setupARView() {
    arView = ARView(frame: view.bounds)
    view.addSubview(arView)
    arView.session.delegate = self
}
```

Anchor 설정

- ARAnchor 생성 및 추가 session.add(anchor:)
- RealityKit Anchor:
  - AnchorEntity(plane:): ARKit이 감지한 평면에 부착
  - AnchorEntity(image:): 트래킹할 이미지 표식에 부착
  - AnchorEntity(world:): 월드 좌표계의 특정 위치에 직접 배치
- Anchor에 컨텐츠(Entity)를 addChild하면 해당 위치에 3D 모델이 고정되어 랜더링 됨.

Anchor 설정

```swift
// 수평 평면 위에 파란 박스 앵커 추가
private func addBoxAnchor() {
    let boxMesh = MeshResource.generateBox(size: 0.1)
    let material = SimpleMaterial(color: .blue, isMetallic: true)
    let boxEntity = ModelEntity(mesh: boxMesh, materials: [material])

    // 수평 평면 감지 Anchor
    let anchor = AnchorEntity(plane: .horizontal)
    anchor.addChild(boxEntity)
    arView.scene.addAnchor(anchor)
}
```

환경 인식 & 조명 추정

- environmentTexturing 설정
- lightEstimate 사용해 실시간 조명 mapping
- DirectionalLight와 EnvironmentResource 적용

조명 추정

```swift
// DirectionalLightEntity를 만들고 앵커에 추가.
private func setupLighting() {
    // 1) 라이트 생성
    let light = DirectionalLight()
    light.light.intensity = 1_000
    light.light.color = .white

    // 2) 회전: 위에서 아래로 비추도록
    light.transform.rotation = simd_quatf(
        angle: -.pi/4,
        axis: [1, 0, 0]
    )

    // 3) 월드 앵커에 붙이고 씬에 추가
    let lightAnchor = AnchorEntity(world: [0, 0, 0])
    lightAnchor.addChild(light)
    arView.scene.addAnchor(lightAnchor)

    directionalLight = light
}
```