

Vision Pro - Tracking Capabilities

What if ...

- The real world could interact with your immersive world?
- You could enrich your surroundings with dynamic / vibrant / artificially living objects?
- Move through your immersive world?
- Create a believable experience as a mixture of real and artificial world where the barriers vanish?

Now great device for it: Vision Pro

→ One key aspect: Good Tracking Capabilities


About me

Matthias Weber

- Freelance developer since 2010
- Swift and (somehow) Objective-C, diverse other languages
- iOS, macOS, visionOS since the beginning, customer and own apps
- But also: other customer projects with e.g. C++ on Windows, Linux
- Several times at Apple Developer Labs for Vision Pro in Munich and London
- Was a research assistant before in the areas HCI, AR/VR and always kept an interest in it
- www.develicious.de

How to do tracking?

Setup Tracking

- Key question: What data do we get as developers? Answered in this talk. 😊
- ARKitSession and DataProviders
- Anchor entities
- SpatialTrackingSession 
- Limited support in simulator, only Anchor Entities and **device transform**

ARKitSession (1)

- Runs an data providers that track specificied object types
- E.g. initialize in an immersive view (or model that gets called from the view)
- @State **var** session = ARKitSession()
- Run with: **try await** session.run([dataProviders])

ARKitSession (2)

- Request authorization on session:
`await session.requestAuthorization(for: [.worldSensing, .handTracking])`
- Monitor session events: `session.events` (authorization and data provider state)
- Documentation: https://developer.apple.com/documentation/arkit/arkit_in_visionos
- **Important:** Only run in an immersive space!! And app has to be focused!

Data Provider

- Data providers are a source of live data from ARKit, work with ARKitSession. (<https://developer.apple.com/documentation/arkit/dataprovider>)
- Always provide anchor updates
- Only provided data providers are supported (can't write your own)
- ImageTrackingProvider, HandTrackingProvider, WorldTrackingProvider, PlaneDetectionProvider, SceneReconstructionProvider
- ObjectTrackingProvider, RoomTrackingProvider, EnvironmentLightEstimationProvider, BarcodeDetectionProvider (Enterprise only), CameraFrameProvider (Enterprise only)



Anchor Entities

- Special entities that you can add to the scene in a RealityView (in contrast DataProviders **don't need it**)
- They anchor to the specified type of tracking (head, hand, world, image, plane, referenceImage, referenceObject, ...)
- `@State var wristEntityCont = AnchorEntity(.hand(.right, location: .joint(for: .wrist)), trackingMode: .continuous)`

SpatialTrackingSession



- Can be used in RealityView, without using ARKit
- Used in combination with AnchorEntity, without using SpatialTrackingSession **no access to transform**
- **let** spatialTracking = SpatialTrackingSession()
let spatialTrackingConfiguration =
 SpatialTrackingSession.Configuration(tracking: [.hand])
await spatialTracking.run(spatialTrackingConfiguration)
- Documentation: <https://developer.apple.com/documentation/RealityKit/SpatialTrackingSession>

Privacy

Usage Descriptions

- No access to camera or sensor data
- Only for certain data
- Request data through usage descriptions
- `NSWorldSensingUsageDescription`, `NSHandsTrackingUsageDescription`
- Authorization types: `worldSensing`, `handTracking` (most providers use `worldSensing`)
- New in 2.0: `cameraAccess`



Image Tracking (1)

- Track static images
- Add them to **a** AR resource group
- Pay attention to Xcode warnings

```
@State var imageInfo = ImageTrackingProvider(  
    referenceImages:  
    ReferenceImage.loadReferenceImages(  
        inGroupNameNamed: "ARReferenceImages"  
    )  
)
```

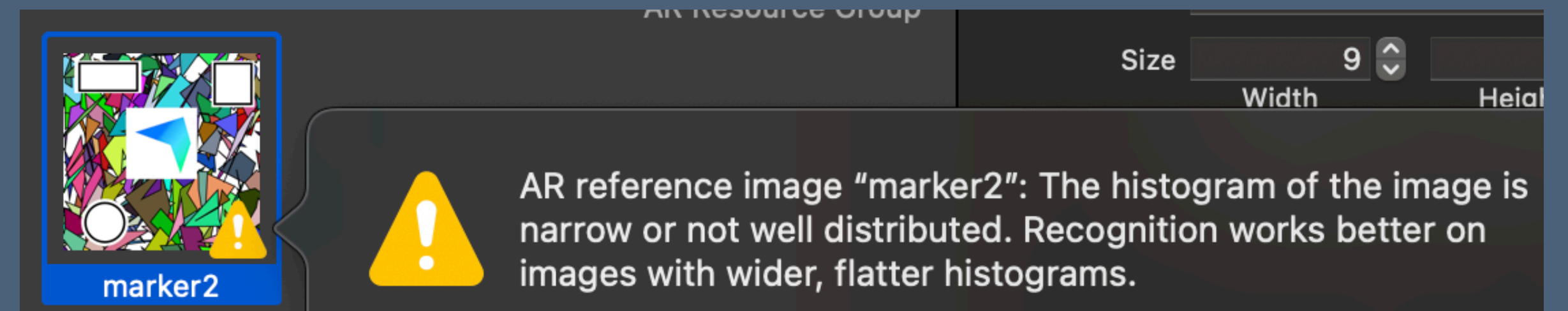
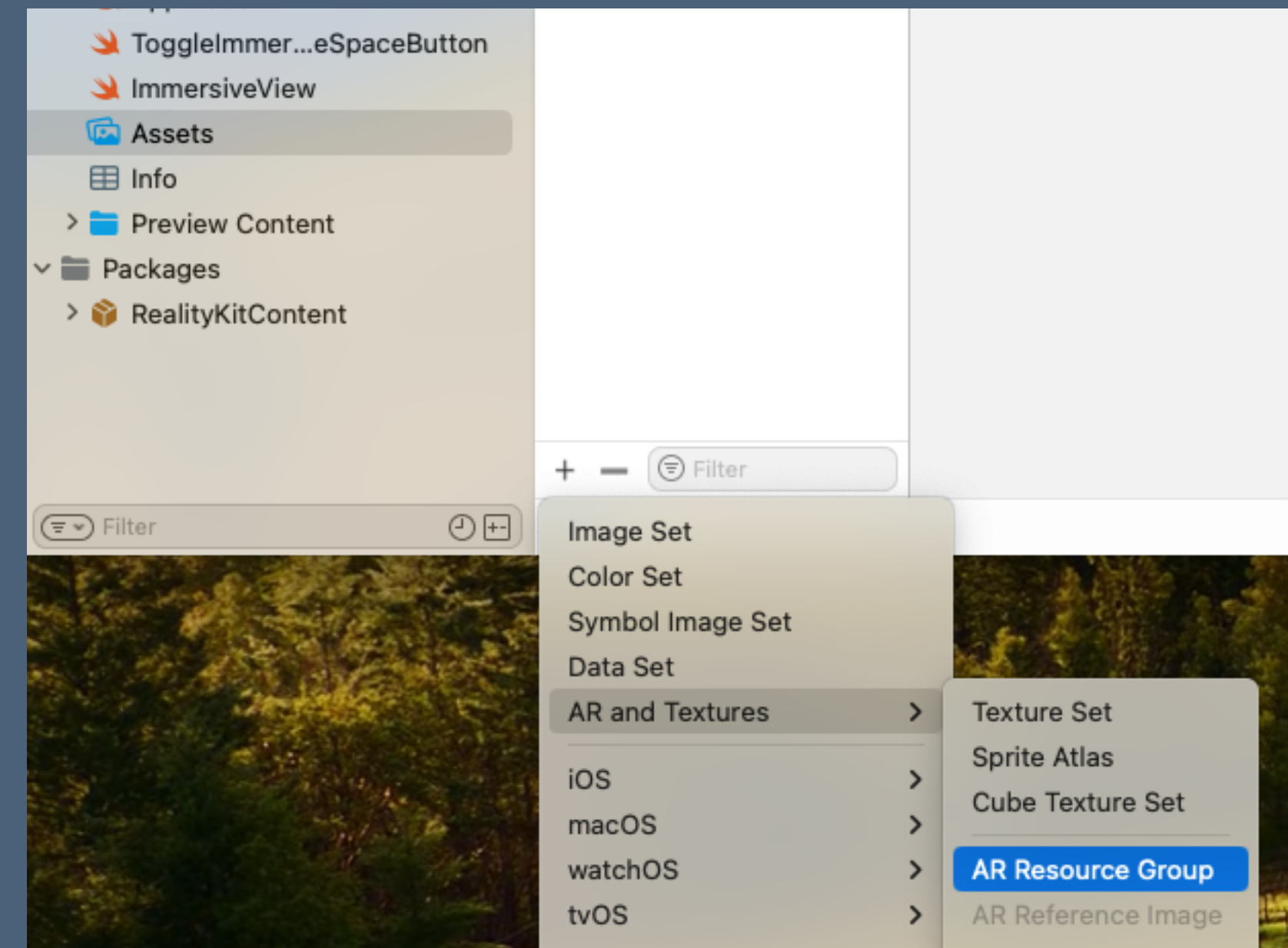
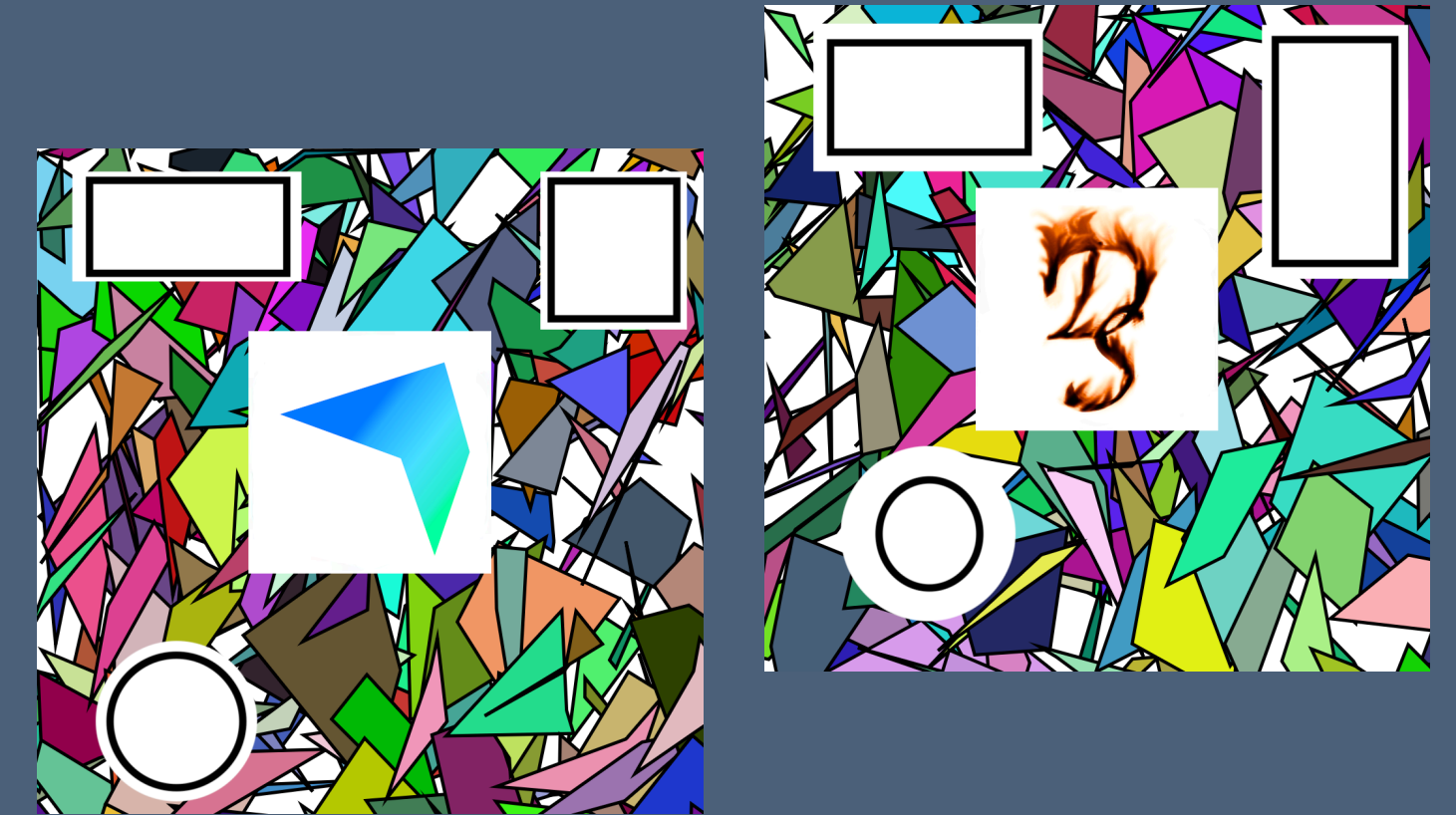
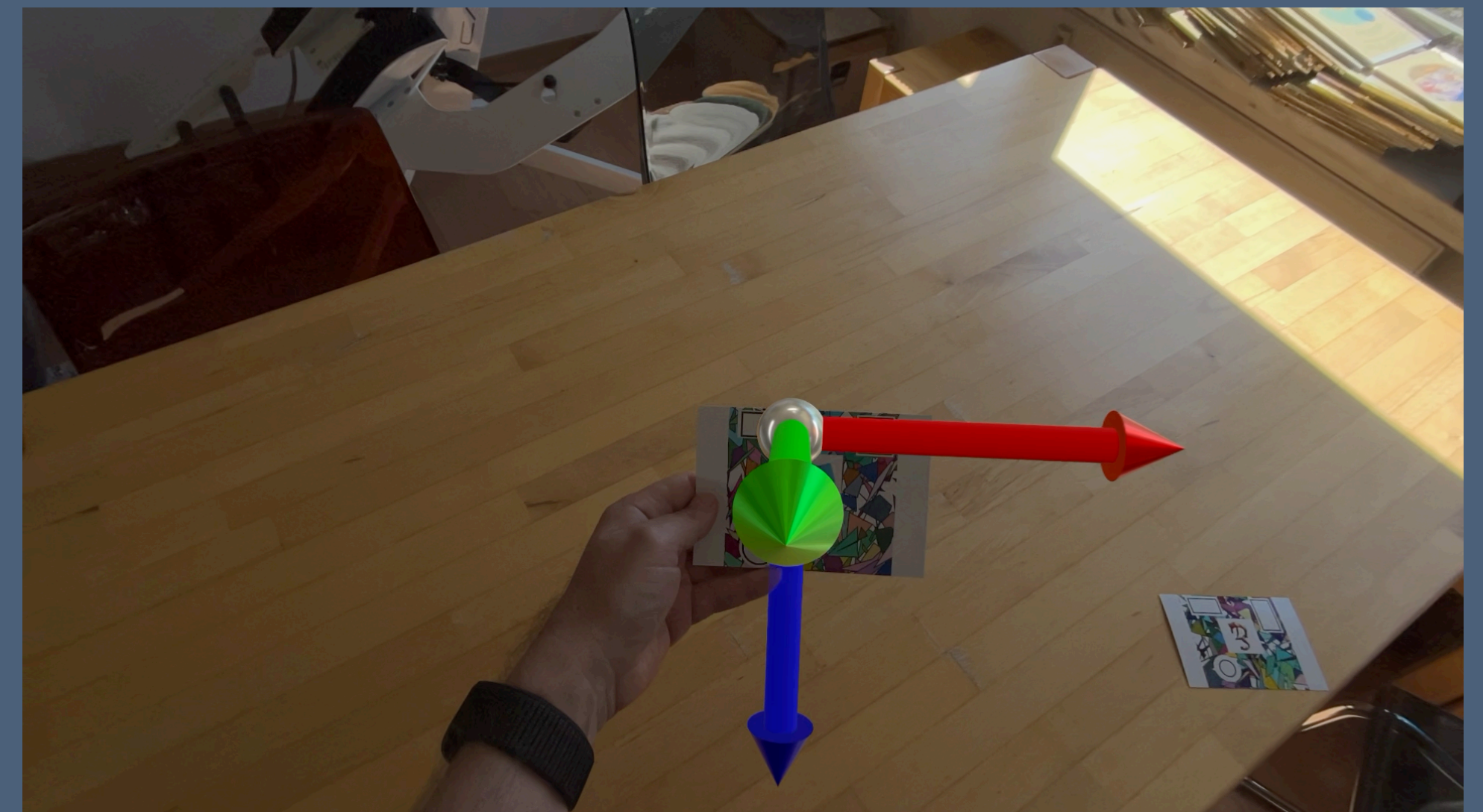
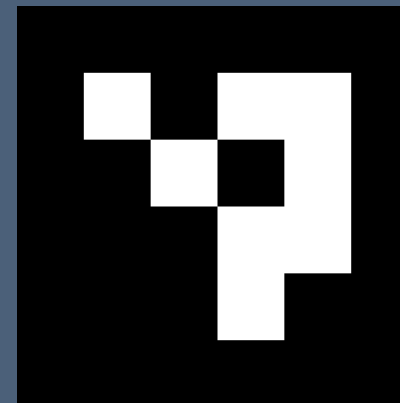


Image Tracking (2)

- Use high contrast images, with color and some geometry
- Taking a photo works, but using a generated one or one that is high quality printed works better, **better no reflection**, provide physical size (!)



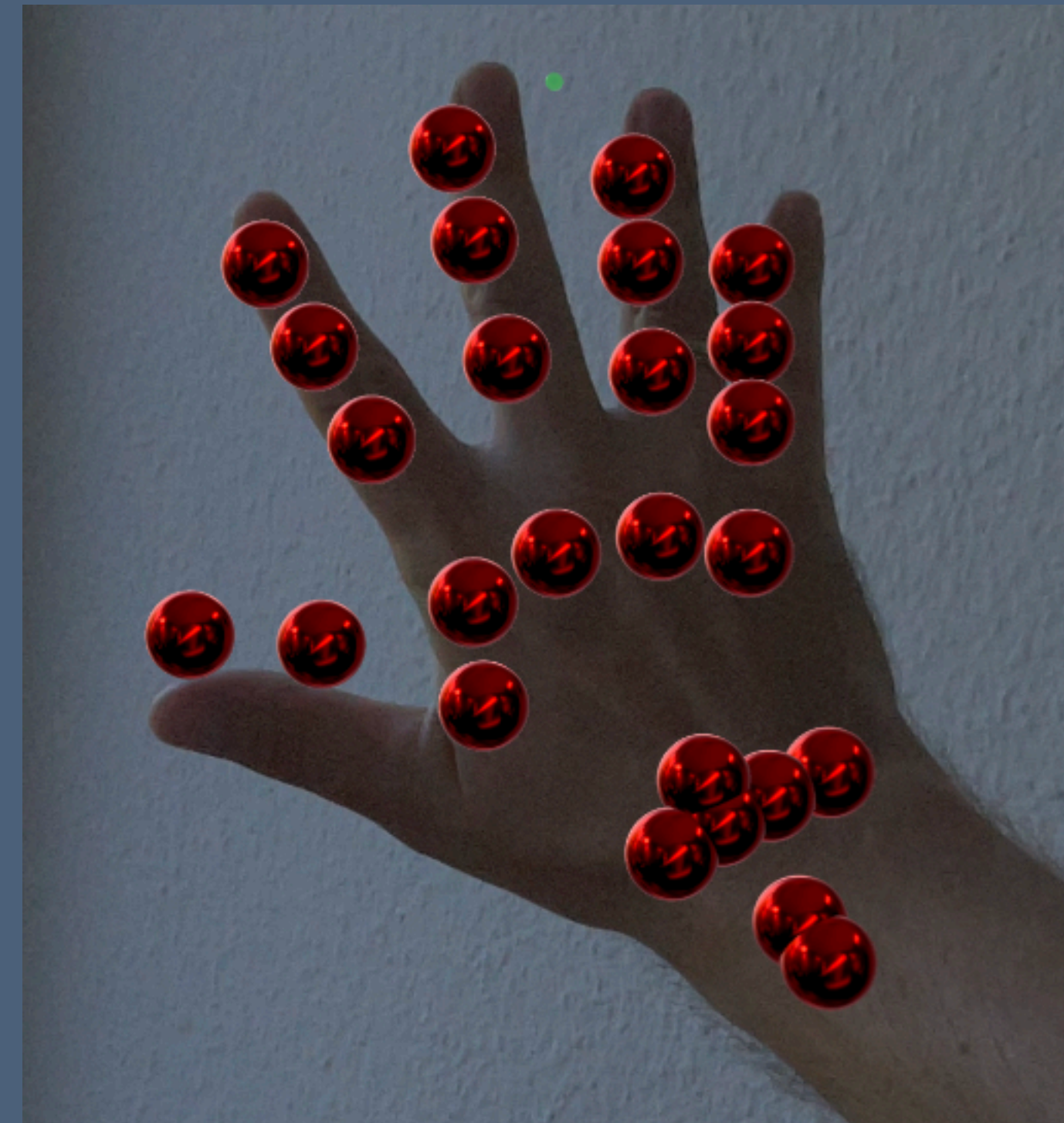
- ARKit type markers do not work well
- Frequency: ~0.5 to ~2s
- Precision: can be around 0.1-1cm



- <https://developer.apple.com/documentation/visionos/tracking-images-in-3d-space>

Hand Tracking (1)

- Track complete hand skeleton of both hands
- HandTrackingProvider: just instantiate, run session, get anchor updates
- Or: Use AnchorEntities
- Authorization: handTracking



Hand Tracking (2)

- Has a certain lag, there is a little jitter, better with visionOS 2.0 (as it is now running at display rate)
- Lag can be avoided by providing a predictive timestamp:
 - `handAnchors(at: timeStamp)` with Compositor Services
 - `AnchorEntity(.hand, trackingMode: .predicted)` with RealityKit

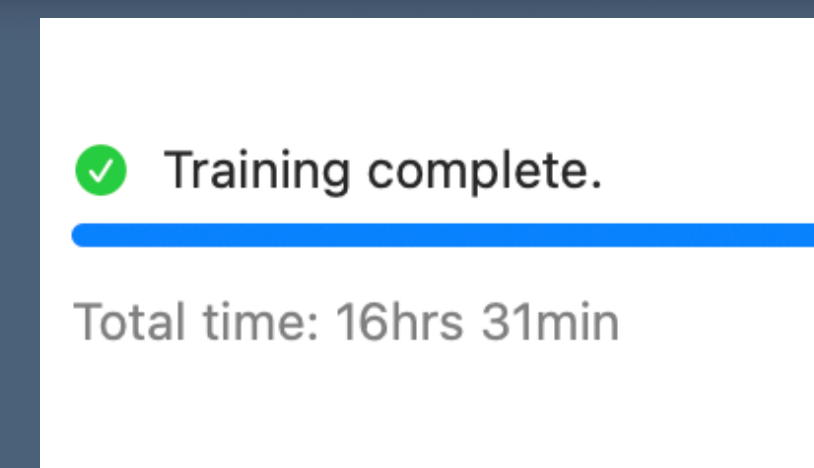
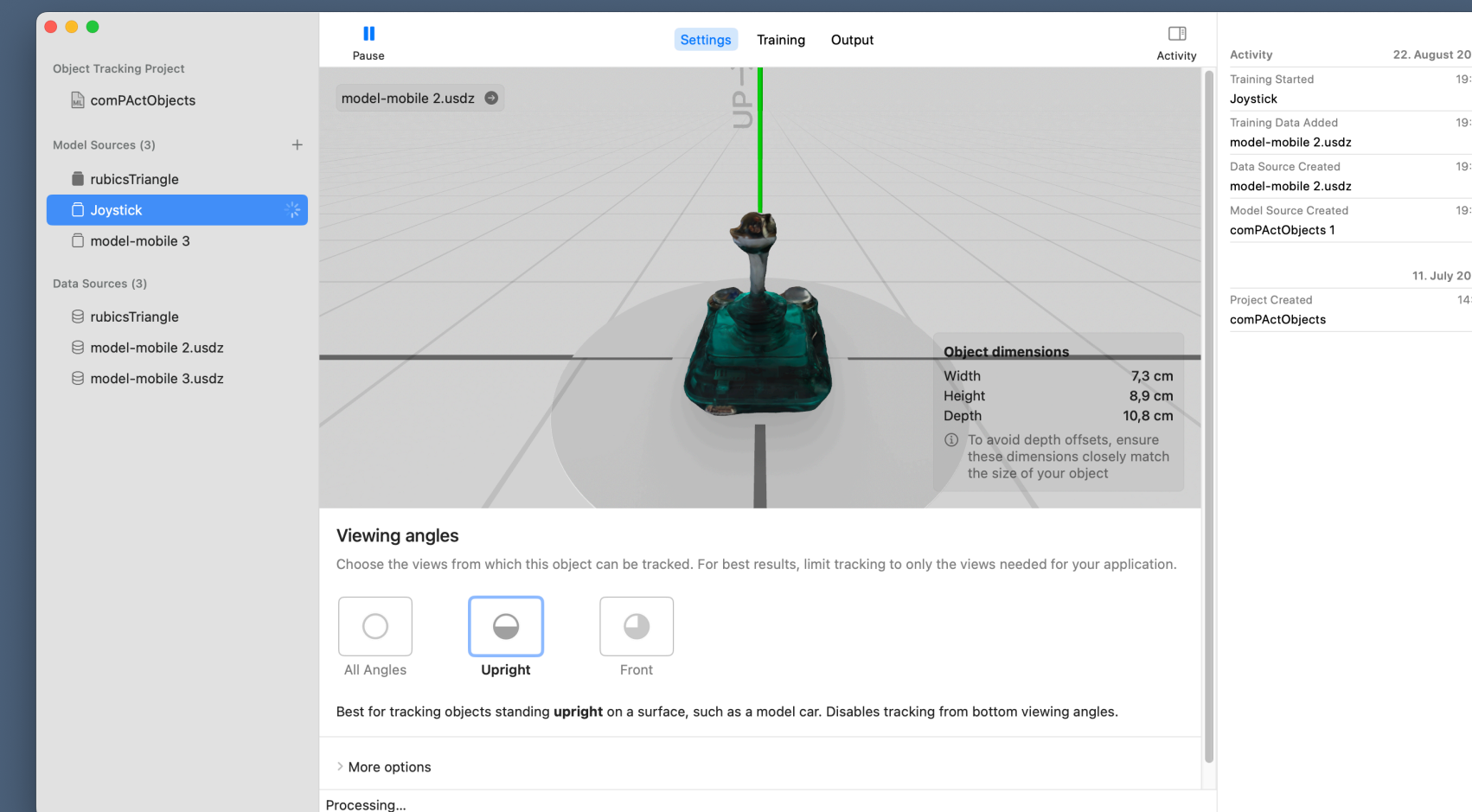
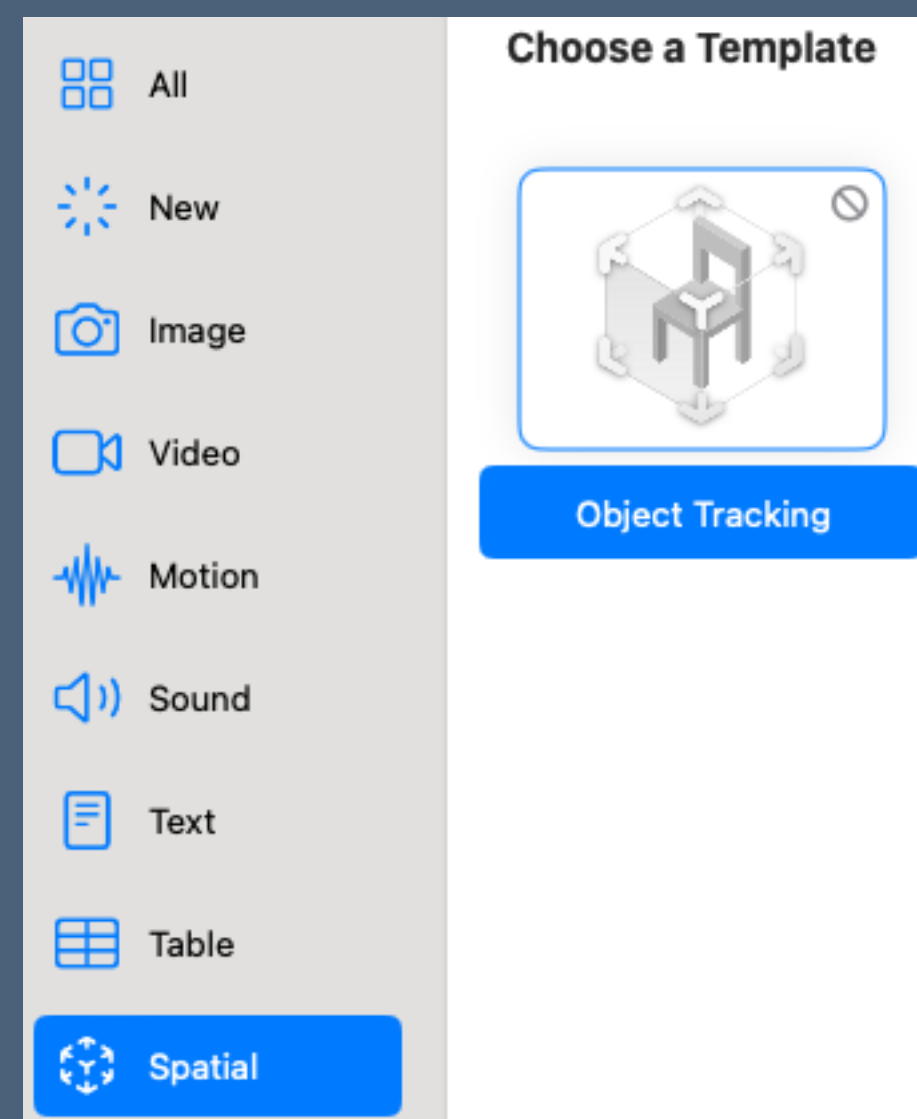


Object Tracking



Train an object model (1)

- Use CreateML (part of Xcode), on macOS Sequoia
- Add a USDZ file to it
- Parameters: Physical **size very** important
- Use orientation parameter



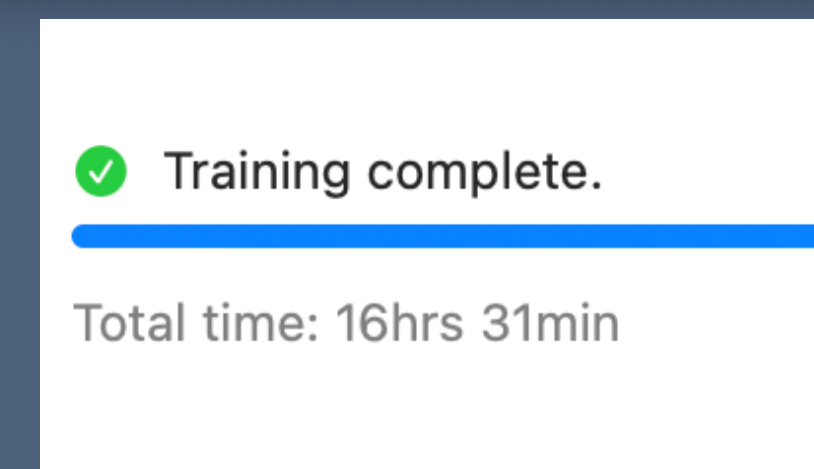
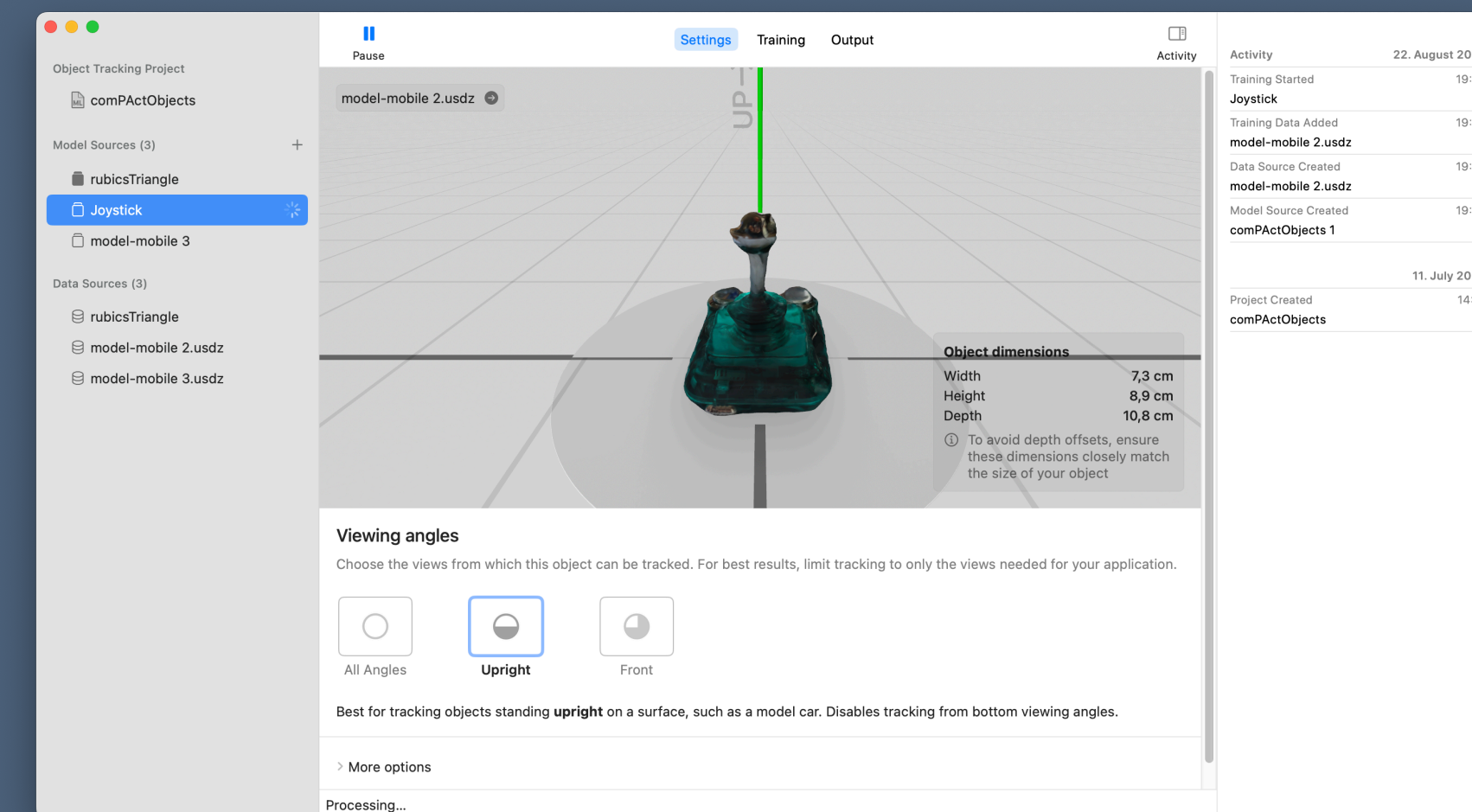
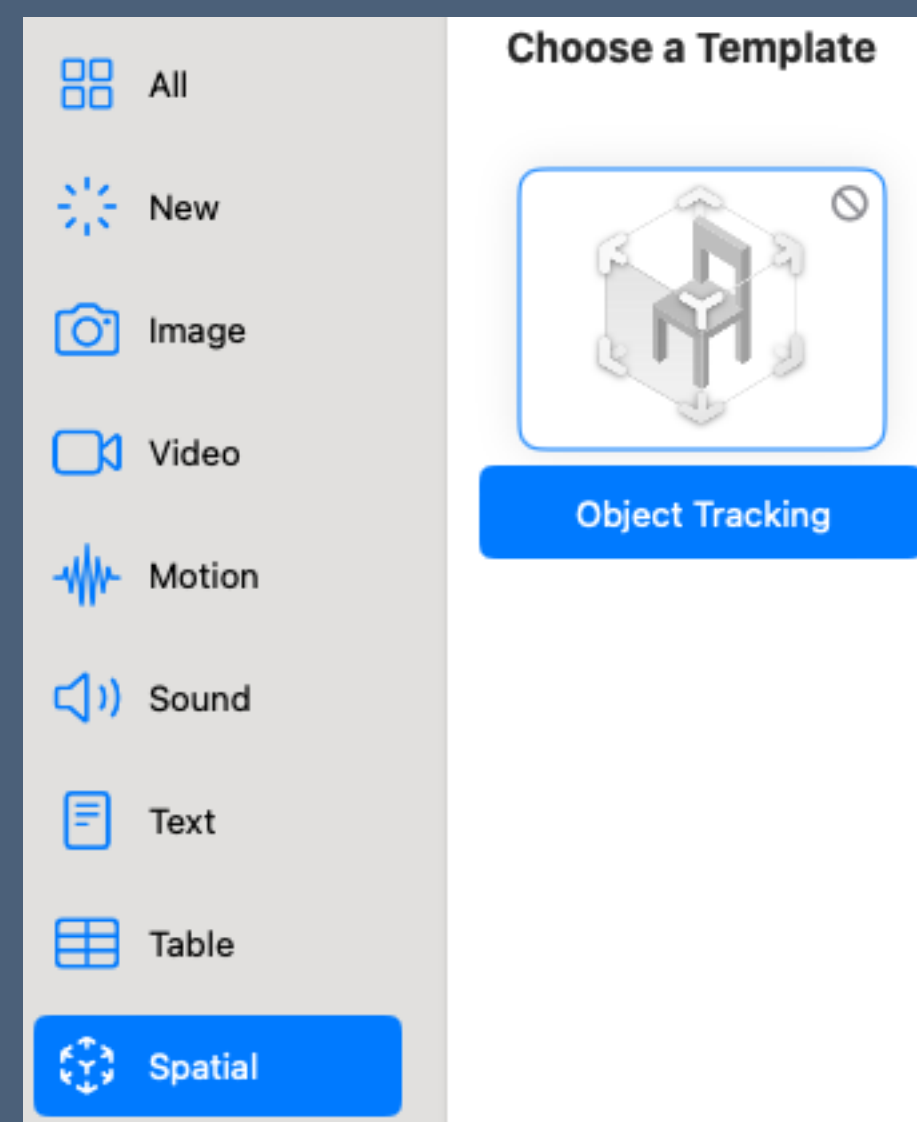
It can take very long!

Object Tracking



Train an object model (2)

- Symmetric objects do not work well, reflecting materials can be difficult (while tracking)
- Either a precise model or a scan
- Had better experience with scanned models
- Can use iPhone for it: Reality Composer (on App Store) or <https://developer.apple.com/documentation/realitykit/scanning-objects-using-object-capture>



It can take very long!

Object Tracking



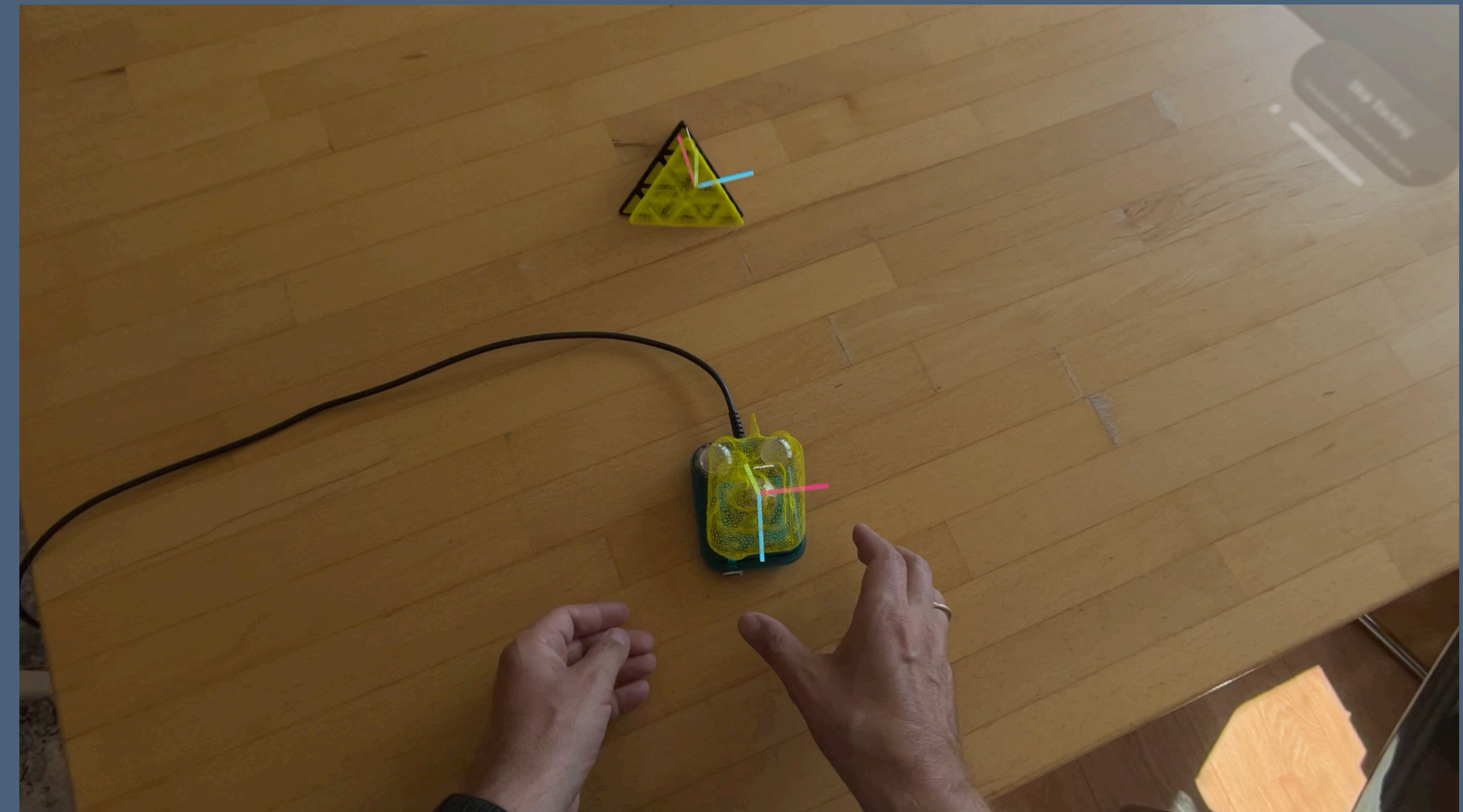
Track an object (1)

- Load reference object:
`try await referenceObject = ReferenceObject(from: url)`
- Provider:
`let objectTracking = ObjectTrackingProvider(referenceObjects: referenceObjects)`
- Get updates by anchor updates **from provider**

Object Tracking

Track an object (2)

- Update rate ok, but not “real-time”
- Capable of tracking even partially occluded objects (e.g. while gripping a handle)
- Uniform/symmetric objects, or objects with a lot of reflection do not work **extremely** well (as documented)
- Sample: https://developer.apple.com/documentation/visionos/exploring_object_tracking_with_arkit



World Tracking

- Place world anchors anywhere and they will stay there
- They will be persisted between sessions
- Precision is as good as with other 3D content placed in the world
- Sample: <https://developer.apple.com/documentation/visionos/tracking-points-in-world-space>
- Apart from that: query device position and orientation (available on simulator as well!)
 - But: No GPS

Plane Detection

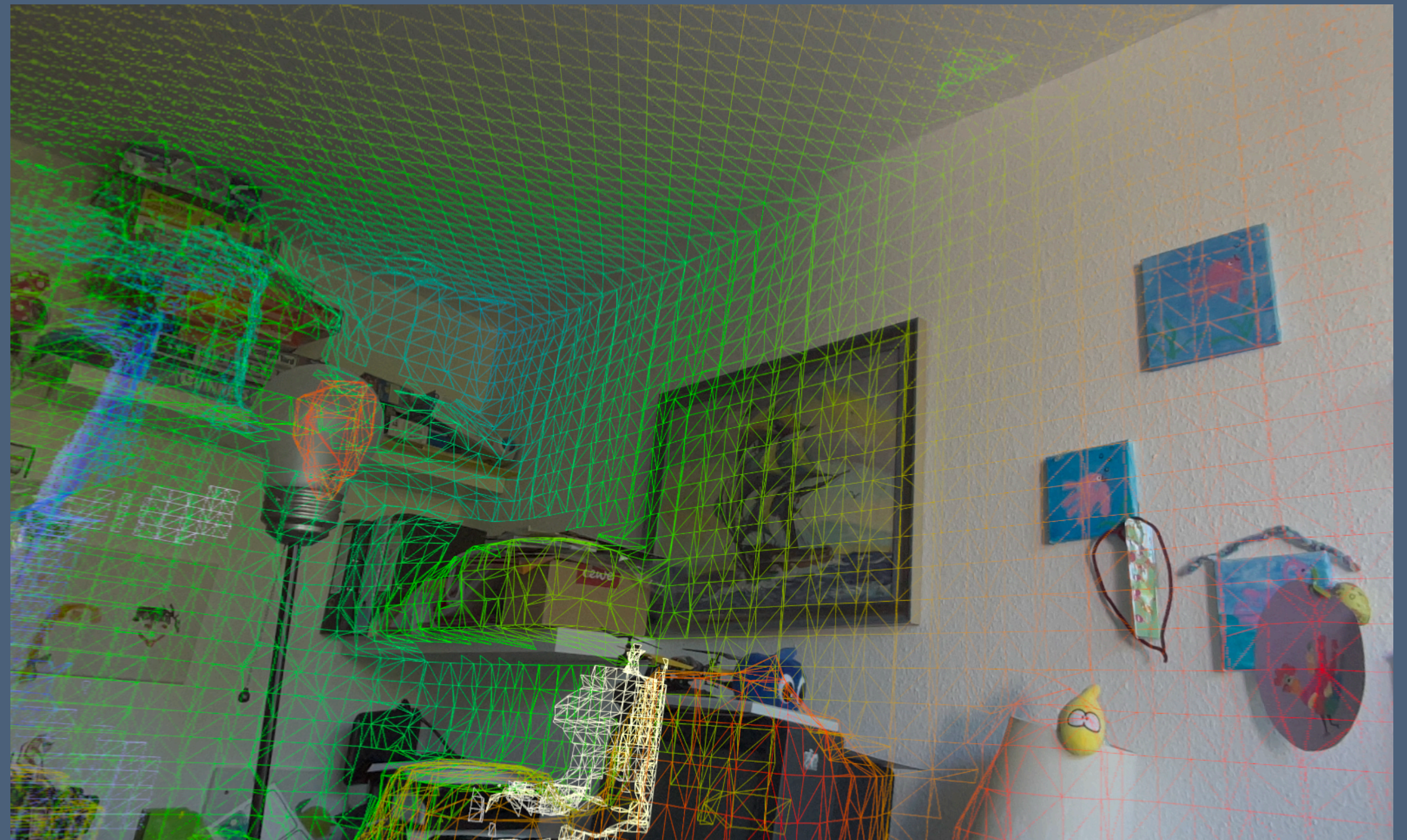
- PlaneDetectionProvider
- Walls, Tables, Floor, Ceiling, Window, Seat
- Provide orientation: horizontal, vertical
- **let** planeData = PlaneDetectionProvider(alignments:
 [.horizontal, .vertical, .slanted])
- Sample: <https://developer.apple.com/documentation/visionos/placing-content-on-detected-planes>



New in
2.0

Scene Understanding (1)

- Mesh reconstruction of surroundings, possible with classification
- **let** sceneReconstruction = SceneReconstructionProvider(modes: [.classification])
- Provides MeshAnchors, these contain the mesh and classification information in buffers
- Currently no segmentation of classification



Scene Understanding (2)

- Buffers: faces (indices as Int), vertices (float3), normals (float3), classification (uchar)
- SIMD3<Float> not same as float3 😊
- Sample: <https://developer.apple.com/documentation/visionos/incorporating-real-world-surroundings-in-an-immersive-experience>

Room Tracking

- Initialize: `let roomTracking = RoomTrackingProvider()`
- `Run` in `ARKitSession`
- `Provides` anchor updates:
 - Provides geometries for current room (similar to `SceneUnderstanding`, but fitted to room)
 - Can check if an anchor is inside the current room
- Sample: https://developer.apple.com/documentation/visionos/building_local_experiences_with_room_tracking

Enterprise APIs (1)



- **Tune object tracking**: more performance vs. better accuracy
- Barcode / QR code scanning
- Camera access
- Only available for internal apps, not App Store apps
- Has to be applied for at Apple

Enterprise APIs (2)



- Documentation: <https://developer.apple.com/documentation/visionos/building-spatial-experiences-for-business-apps-with-enterprise-apis>
- WWDC24 session 10139: Introducing enterprise APIs for visionOS: <https://developer.apple.com/wwdc24/10139/>

Conclusion

- Young platform that still has to evolve, especially for update rates
- But: A lot of tracking possibilities, some with very good quality
- Apple works hard to improve it

Thank you very
much!

Questions?