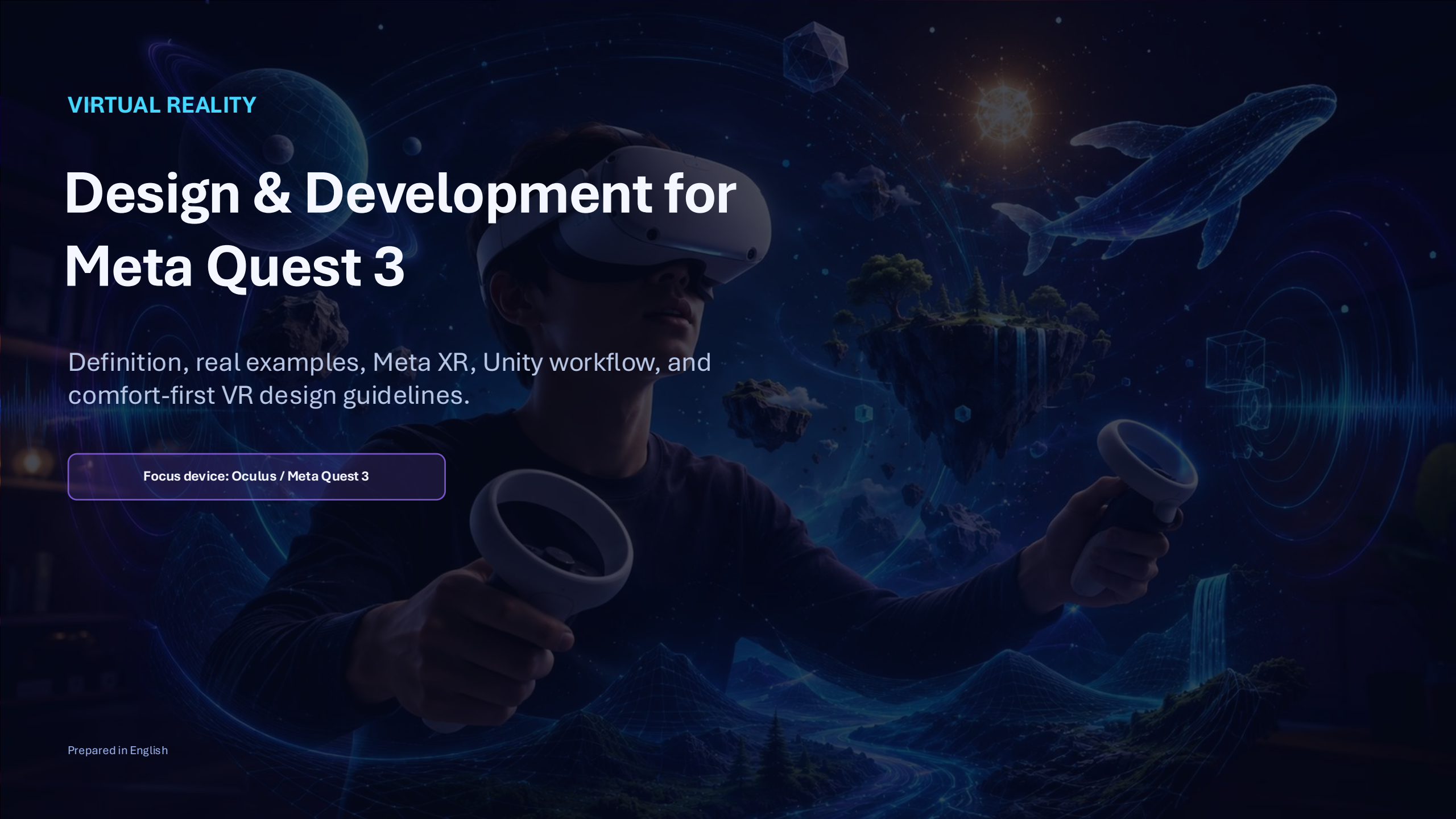
VIRTUAL REALITY

Design & Development for Meta Quest 3

Definition, real examples, Meta XR, Unity workflow, and comfort-first VR design guidelines.

Focus device: Oculus / Meta Quest 3



Prepared in English

What is Virtual Reality?

VR is an interactive, computer-generated 3D environment that surrounds the user and responds to head, hand, and body movement.

In practice, a headset replaces the user's normal field of view with stereoscopic images, spatial audio, tracked controllers or hand tracking, and real-time rendering.

Presence
The feeling of "being there" inside the simulated space.

Embodiment
Head and hand movement are mapped into the virtual world.

Interaction
Users manipulate objects, menus and environments directly.

Sources: Britannica VR definition; Ada Lovelace Institute immersive technologies explainer.

Where VR is used

From “watching a screen” to “acting inside a space”.

Games & entertainment
Beat Saber-style rhythm, immersive worlds, social VR events

Training & simulation
Safety drills, emergency procedures, medical or aviation practice

Education & science
Anatomy, museums, labs, field trips impossible in real life

Design & collaboration
Architecture walkthroughs, product prototyping, remote meetings

Therapy & wellbeing
Exposure therapy, rehabilitation, mindfulness environments

VR, AR and MR: one continuum

Quest 3 can run fully virtual experiences and also mixed-reality scenes through color passthrough, scene understanding and anchors.

Physical world Passtrough vision

AR overlay

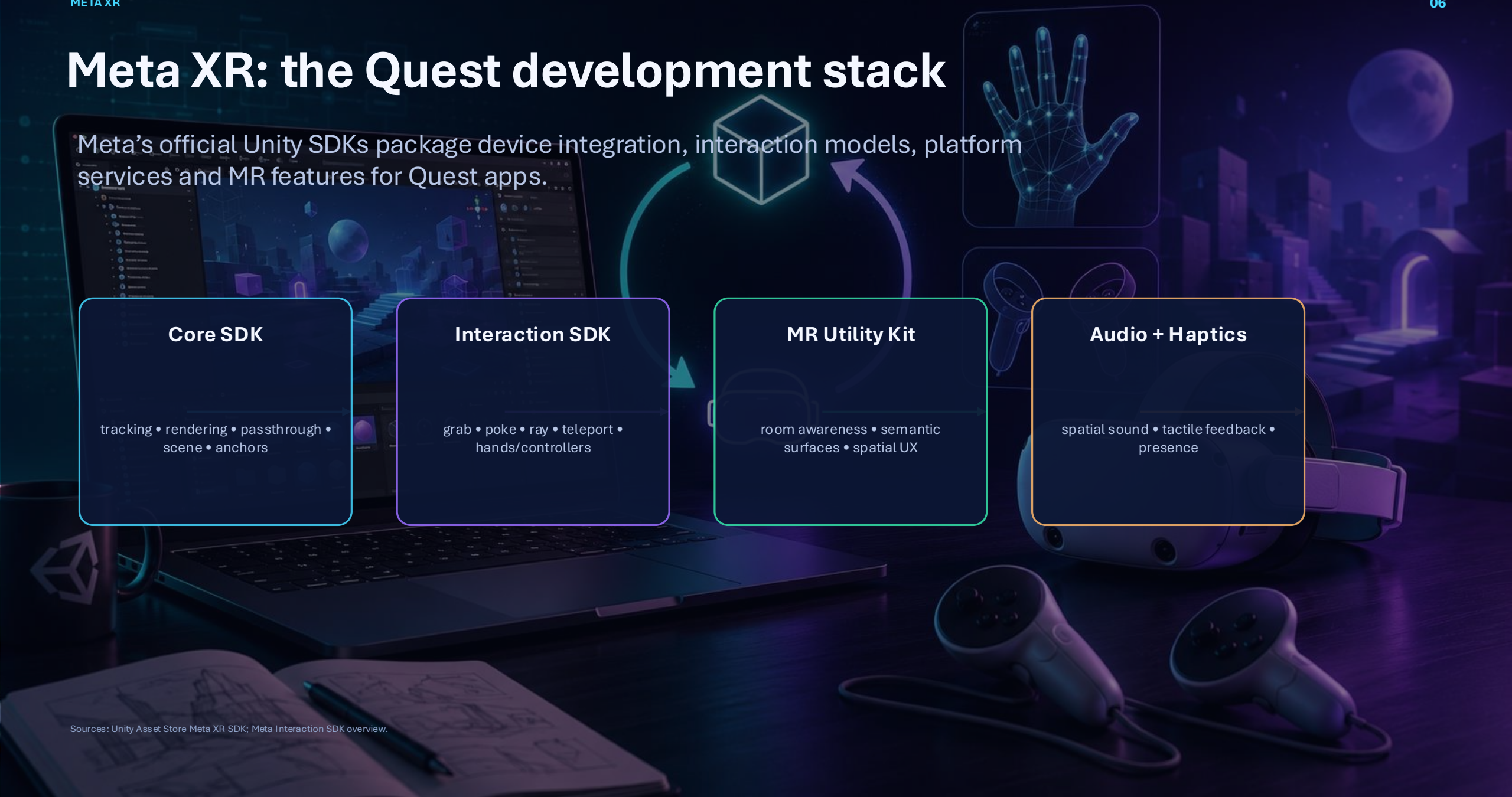
Mixed Reality

Virtual Reality

Sources: Meta Horizon OS design overview: passthrough, scene understanding, spatial anchors, locomotion.

Meta XR: the Quest development stack

Meta’s official Unity SDKs package device integration, interaction models, platform services and MR features for Quest apps.



Core SDK

tracking • rendering • passthrough • scene • anchors

Interaction SDK

grab • poke • ray • teleport • hands/controllers

MR Utility Kit

room awareness • semantic surfaces • spatial UX

Audio + Haptics

spatial sound • tactile feedback • presence

Sources: Unity Asset Store Meta XR SDK; Meta Interaction SDK overview.

Unity workflow for Quest VR

Unity is the scene editor, real-time engine, asset pipeline, and build system. The Quest app is ultimately an Android/OpenXR application.

1 Project setup

XR Plug-in Management, OpenXR/Meta packages, Android build target

2 Scene + scale

1 Unity unit = 1 meter; design around real human reach

3 Origin

Camera rig, tracking space, controllers and hands

4 Interactions

XRI or Meta Interaction SDK: grab, poke, ray, UI canvas

5 Build & test

Profile on device, iterate comfort and performance

Sources: Unity XR Interaction Toolkit docs; Meta Unity app development documentation.

Comfort first: the non-negotiables

VR can feel magical only when the user's body believes the experience is stable, responsive and under control.

Keep motion predictable
Avoid forced camera movement, surprise acceleration, or artificial camera shake.

Maintain frame rate
Dropped frames increase latency and can quickly break comfort. Optimize early on Quest hardware.

Provide comfort options
Offer teleport / snap turn / vignette / seated mode rather than one locomotion style.

Respect the body
Use real-world scale, stable horizons and interactions within comfortable reach.

Sources: Meta Comfort and Locomotion design guidelines; Unity Tunneling Vignette Controller docs.

Headset-friendly UI design

Design spatial UI as objects in 3D space, not as a flat mobile screen glued to the face.

Place UI at comfortable distance
Avoid panels too close to the eyes; keep main content in front and slightly below eye level when possible.

Use large, high-contrast text
Favor short labels, clear hierarchy and readable interaction targets.

Avoid “head-locked” UI overload
Use world-locked or hand-attached menus for most tasks; reserve head-locked UI for critical status only.

Use depth deliberately
Do not force users to refocus rapidly between near and far elements.

Support seated and standing users
Menus must work with different heights, arm reach and play spaces.

Sources: Meta Horizon OS design overview: display, components, input hierarchy, hand representation.

Locomotion (Movement types that reduce motion sickness)

Teleport

Teleportation + room-scale walking

Head Rotation

Snap turns; avoid smooth rotation by default

Vehicle comfort

Cockpit frames and stable references reduce vection

Advanced option

Smooth locomotion only with settings: speed, vignette, comfort mode

How users interact inside VR

VR interaction is embodied: the user expects objects to react like physical things, but with clear digital feedback.

Direct grab
near objects with hand/controller pose

Ray pointer
far UI, menus, object selection

Poke / press
buttons, sliders, toggles

Hand tracking
natural gestures, direct manipulation

Haptics + audio feedback → output
confirmation, affordance and presence

Sources: Meta Interaction SDK overview; Meta Horizon OS design overview input primitives/controllers/voice/haptics.

Implementation checklist for comfort

Translate design principles into actual settings and components before playtesting.

Locomotion

- TeleportationProvider
- SnapTurnProvider
- optional: ContinuousMove with Tunneling/Vignette for smooth move

Interactions

- XR Interaction Manager
- Ray + Direct Interactors
- Meta hands/controllers rig
- visible affordances + haptics

UI

- World-space canvas
- laser pointer + poke targets
- large hit areas
- no tiny text or edge-only UI

Performance

- profile on Quest 3
- fixed foveated rendering when appropriate
- batching/LOD/occlusion
- sustain target refresh rate

https://developers.meta.com/horizon/documentation/unity/unity-isdk-teleport-interaction/
https://developers.meta.com/horizon/documentation/unity/unity-isdk-grabbing-objects/
https://developers.meta.com/horizon/documentation/unity/unity-isdk-uiset/

Key sources used

https://developers.meta.com/horizon/documentation/unity/unity-tutorial-hello-vr/

Meta Quest 3
Meta Quest 3 official product page; VR-Compare Quest 3 specifications.

Meta XR stack
Meta Horizon OS Developers: Interaction SDK overview; Unity Asset Store: Meta XR SDK.

Design guidelines
Meta Horizon OS Developers: Comfort and immersive experience design overview.

Unity implementation
Unity XR Interaction Toolkit documentation; Unity Tunneling Vignette Controller.

Use this deck as a practical introduction: validate all design decisions with real users wearing Quest 3, not only inside the Unity editor.