Unity: Designing Extended Reality (XR) Applications

What is XR?

XR is an emerging technology (aka. spatial computing) that:

- Blends physical and digital worlds
- Create multimodal, spatial, and immersive experiences and applications

XR is multimodal because it blends visual, haptic, and auditory interactions that are realistic, as far as possible.

XR is also spatial because most interactions are anchored in physical and virtual spaces and environments.

And finally, XR is immersive because it creates realistic and believable worlds that blur boundaries between the real and the digital world.

As an emerging technology, XR has its toolsets worth learning and mastering.

XR is a Spectrum of Three Realities

XR is an umbrella term that includes three types of realities:

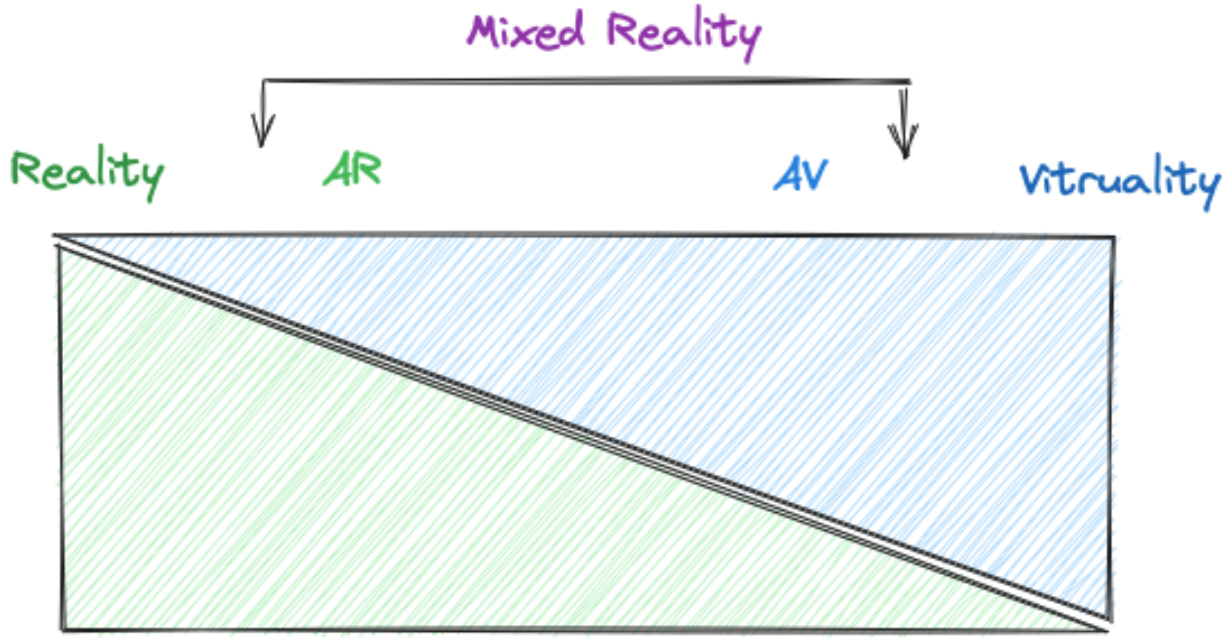
- ***augmented reality*** (AR),
- ***virtual reality*** (VR), and
- ***mixed reality*** (MR).

We often get asked what's the difference between these three realities.

There is a subtle difference between these technologies. However, the field of XR is still developing and boundaries are a bit blurred.

XR is a Spectrum of Three Realities

- AR: Layers virtual content over a user's real environment
- VR: Simulates a self-contained virtual environment around the user
- MR: Blends virtual and user's real environment with interaction between both



What is AR?

AR technology overlays digital content (texts, graphics, images, videos, audio, 3D objects, etc.) over the user's view of the real environment so that the digital content is seen as part of the real environment. The more the digital content blends with the real world the more engaging the AR content can be.

Hand-held AR

AR technology is the one promising in the XR spectrum because of its simplicity. It does not require headsets or goggles.

Modern smartphones, tablets, and PCs are equipped with capabilities to handle AR experiences and applications. Millions of users around the world have one or more of these types of devices. That’s why some of the famous AR applications use hand-held devices.

- Pokemon-go
- Ikea Place
- Snapchat

Head-worn AR

This type of AR projects digital content into a headset while enabling users to see-through (pass-through).

It is suitable for immersive experiences or where the user needs to perform tasks with their hands. For example, Meta Quest 2 or 3 and Hololens are examples of head-worn AR devices.

Head-up AR

Head-up AR project digital content into the windshield. For example, in automotive or fighter jets the driver or pilot will have information displayed in their view.

What is VR?

VR technology creates an isolated virtual world around the user through a headset. The user is immersed in the virtual environment and can explore, navigate, and interact with it.

Unlike AR, VR does not have a see-through or pass-through.

What is MR?

MR technology blends virtual and real content enabling users to interact with virtual as if they were in the real world. MR is the best of both worlds of AR and VR.

XR Coordinate System

XR toolkits place virtual objects in the real world so that they look like real objects. To do so, toolkits, rely on **spatial coordinate systems**.

All objects we add to an XR scene have XYZ coordinates. These are cartesian coordinate systems, which can be right-handed or left-handed (Unity is left-handed).

XR Spatial Mapping

Spatial mapping involves using sensors, cameras, or other technologies to capture the geometry, structures, surfaces, furniture, doors, features, and layout of a physical space. XR applications that require spatial mapping apply room scanning to collect to create a digital representation of the environment.

XR Spatial Anchors

A spatial anchor is a point in the physical world that is mapped and tracked in both the real world and the virtual world.

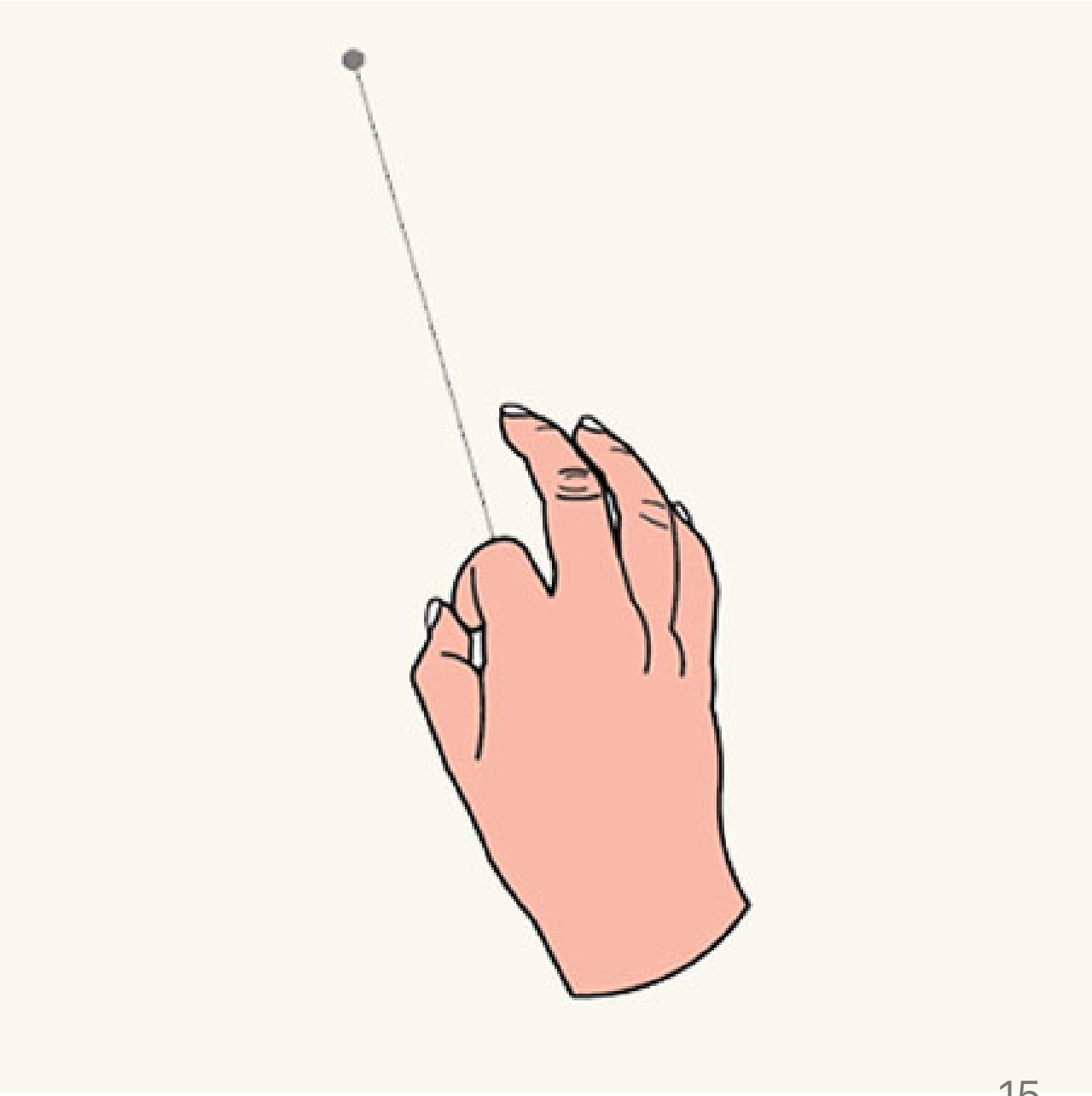
Anchors are important to place virtual content in specific locations in the physical environment. This ensures that when users move their devices or change positions, the virtual content stays in place in the real world.

XR Scene Understanding

XR scene understanding combines spatial mapping and computer vision techniques to process and understand details in the environment. This enables users to interact intelligently with the real world.

XR Rays

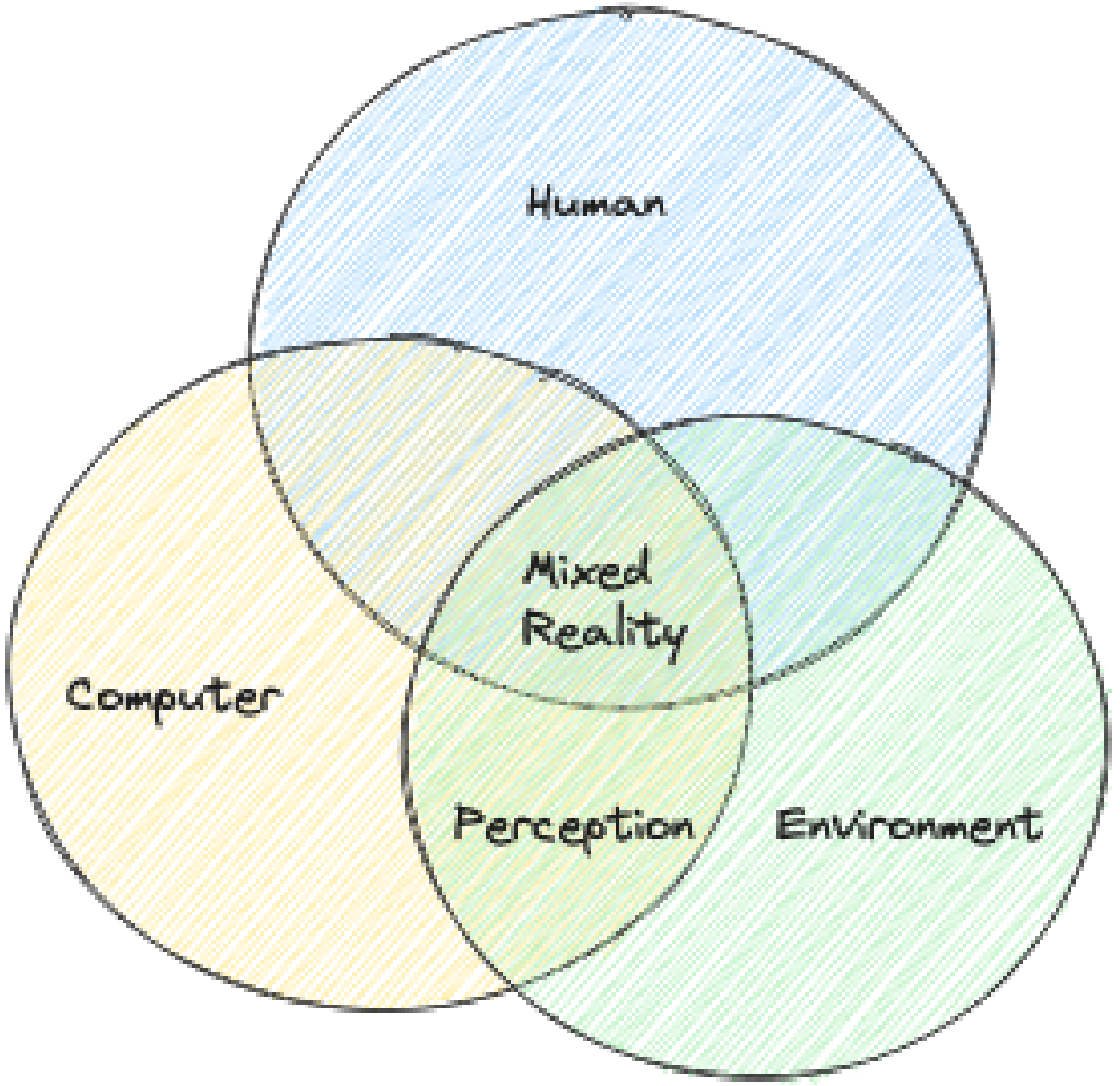
Ray is an XR technique that draws a ray out from the center of the user's palm or controller. Rays are extensions of users’ hands.



XR Interaction

XR is characterized as human-computer-environment interaction.

- Human understanding: capturing human interactions and input, including, position, hand-tracking, eye-tracking, and speech
- Environment understanding: mapping and anchoring of spaces, surfaces, locations, and objects
- Computer processing: sensing, rendering, and keeping track of both human and environment



XR Interaction Models

No matter how beautiful an XR app is, it can be useless without interaction.

- Hand
- Eye
- Voice

The interaction models are mental models to allow users to manipulate objects in XR to complete tasks and workflows. Overall, there are four interaction models

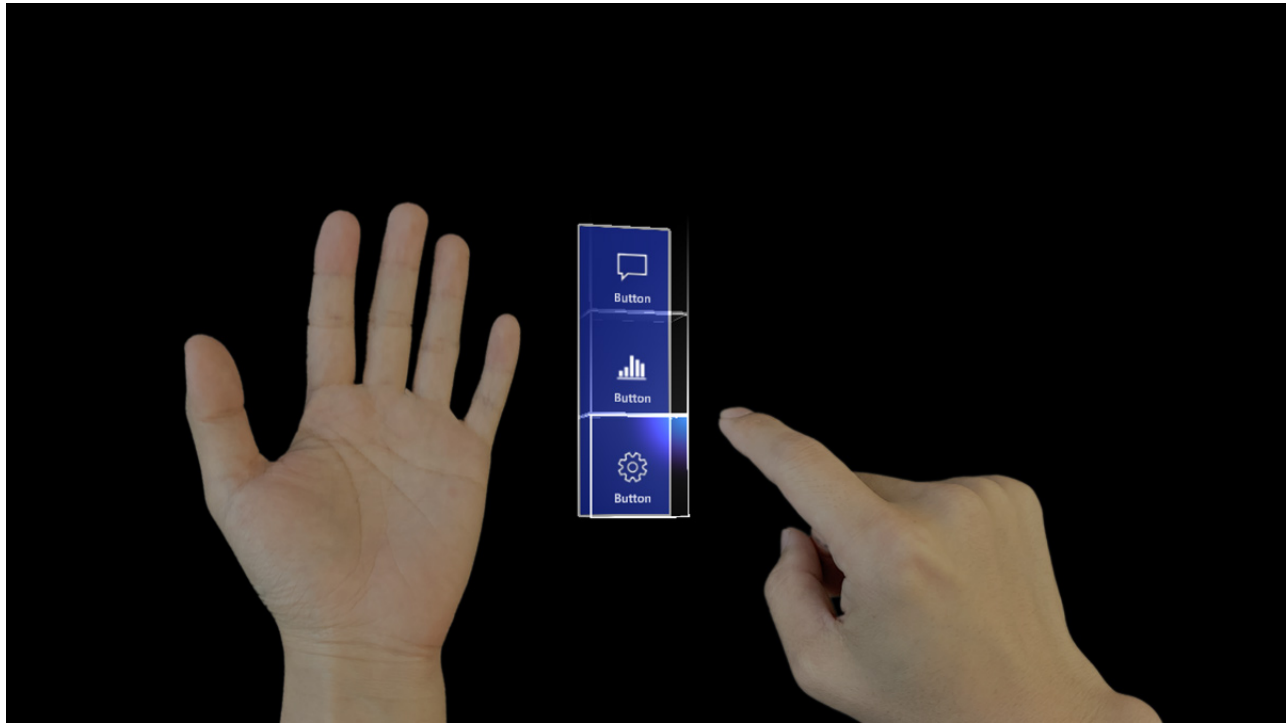
- Hand-based
- Motion controller-based
- Voice-based
- Gaze-based

XR Hand-based interaction

- XR Direct Manipulation
- XR Point and Commit

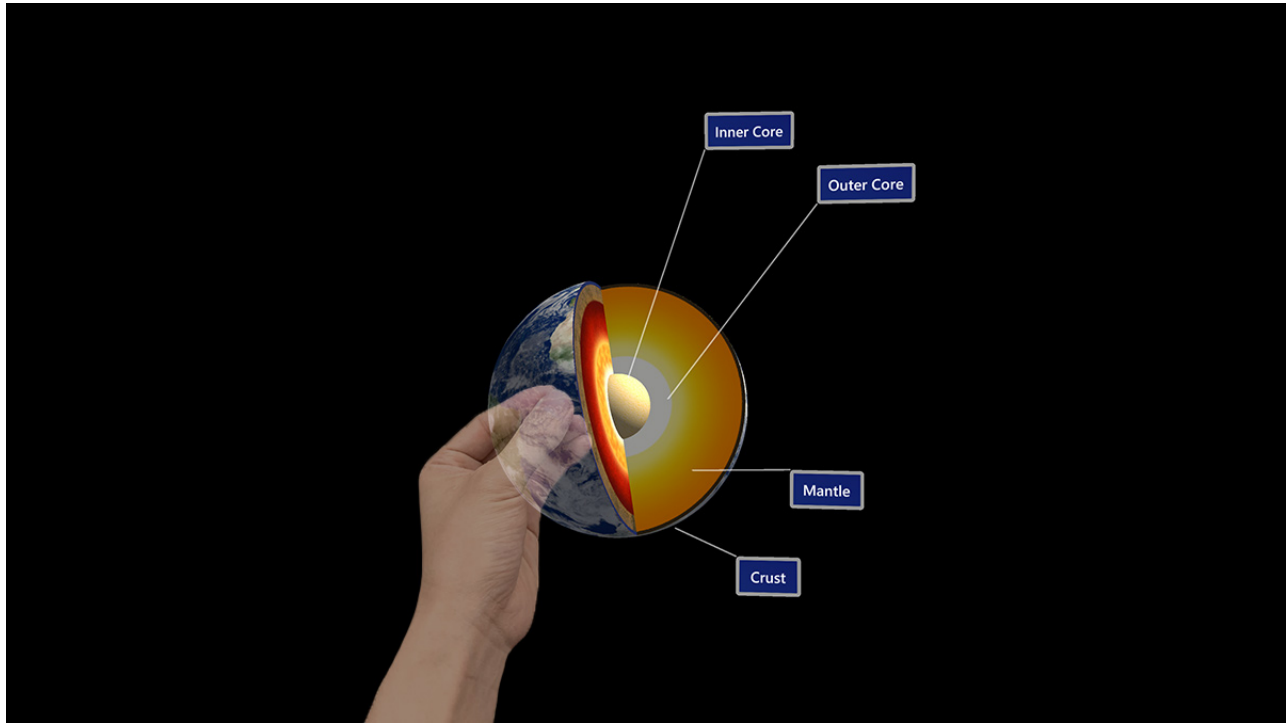
XR Hand Menu

A technique that allows the user to quickly bring up hand-attached UI. It is accessible anytime. It can be shown and hidden easily and is great for quick actions.



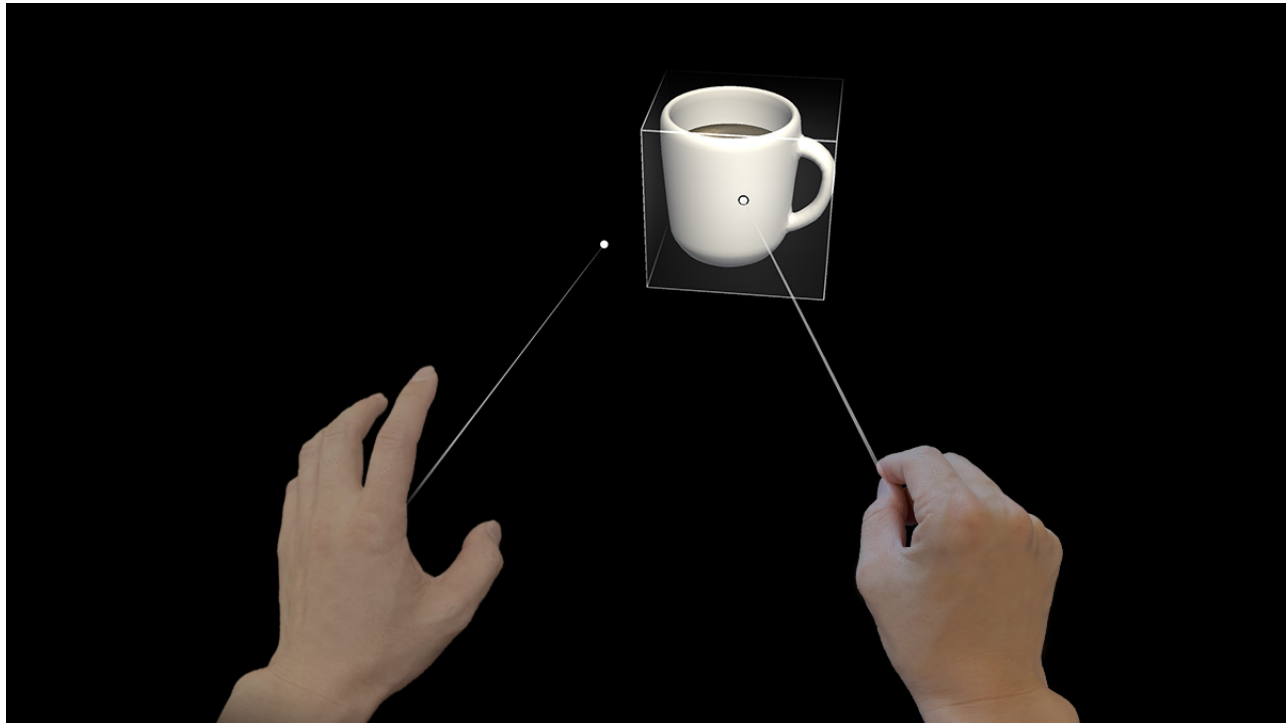
XR Direct Manipulation

Direct manipulation is an input model where users use their hands to interact with objects in an XR scene. This is referred to as “near” interaction. This model mimics real-world interactions. For example, a user can poke a button to click it, grab and rotate an object in 3D space, or point and scroll a view.



XR Point and Commit

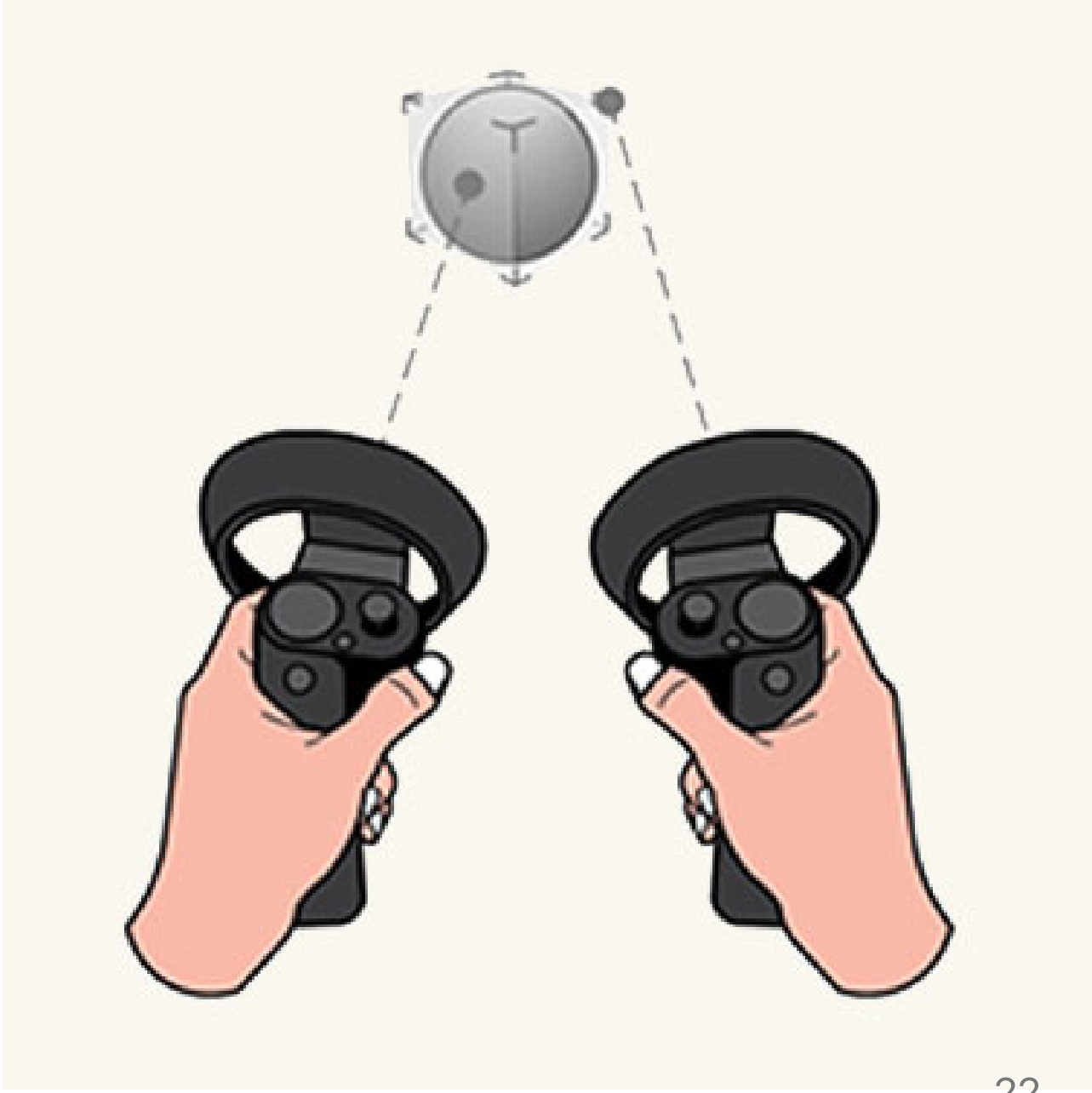
Point and commit an XR model that allows users to target, select, and manipulate **out-of-reach** objects in XR. It is also called “far” interaction. It is unique to XR as we don’t interact with objects this way in the real world. It is a two-state interaction, where first user points with hands (or controllers) and then commits to perform an interaction.



XR Motion Controllers

Motion controllers are hardware accessories that allow users to take actions in a 3D XR space. This includes:

- Manipulating objects
- Locomotion
- Teleportation



XR Hand-free model

This model can be practical for some scenarios. For example, users' hands can be busy, hand fatigue or disability, or tight spaces. In this case, voice- and eye-based interactions can be useful.

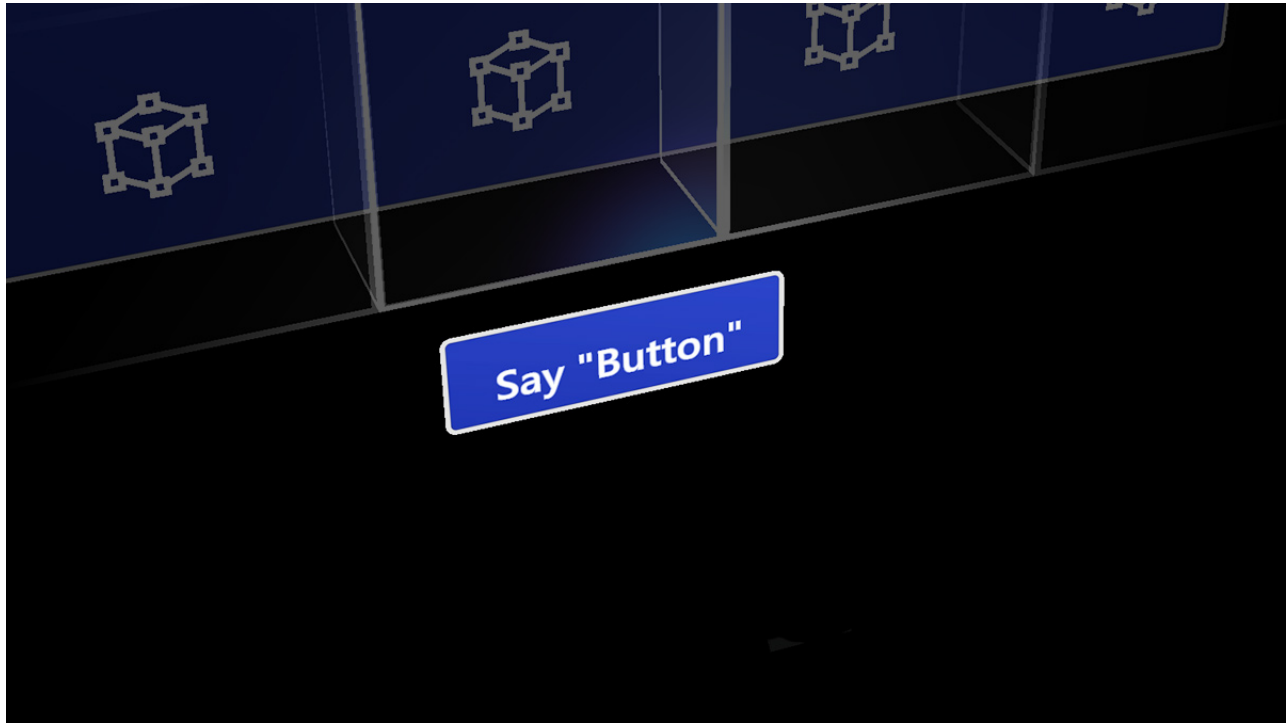
- XR Voice Interaction
- XR Gaze-based Interaction

XR Voice Interaction

This model is part of the XR **Hand-free model**. This can be seen as a "*see it, say it*" interaction where the user can interact with objects and elements of the scene by talking out loud.

Similar to XR Point and Commit, the head or the Gaze (see **XR Gaze-based Interaction**) is the means to target objects. Some interactions might not require targeting.

Voice interaction might not be practical in louder conditions.



XR Gaze-based Interaction

This model is also part of **XR Hand-free model**. This includes eye-gaze and head-gaze interaction.

Eye-gaze enables the use of a user's eyes as an input method for interacting with objects in XR.

It can rely on eye movement (saccades, smooth, and fixations) and/or gaze direction (direction of where a person is looking).

XR Gaze-based Interaction

There are three types:

- Gaze and commit
- Gaze and dwell
- Head-gaze and commit

XR Gaze-based Interaction

Gaze and commit is a type of interaction where

- a user looks at an object (targeting), then
- perform a secondary input, such as using hand or voice

XR Gaze-based Interaction

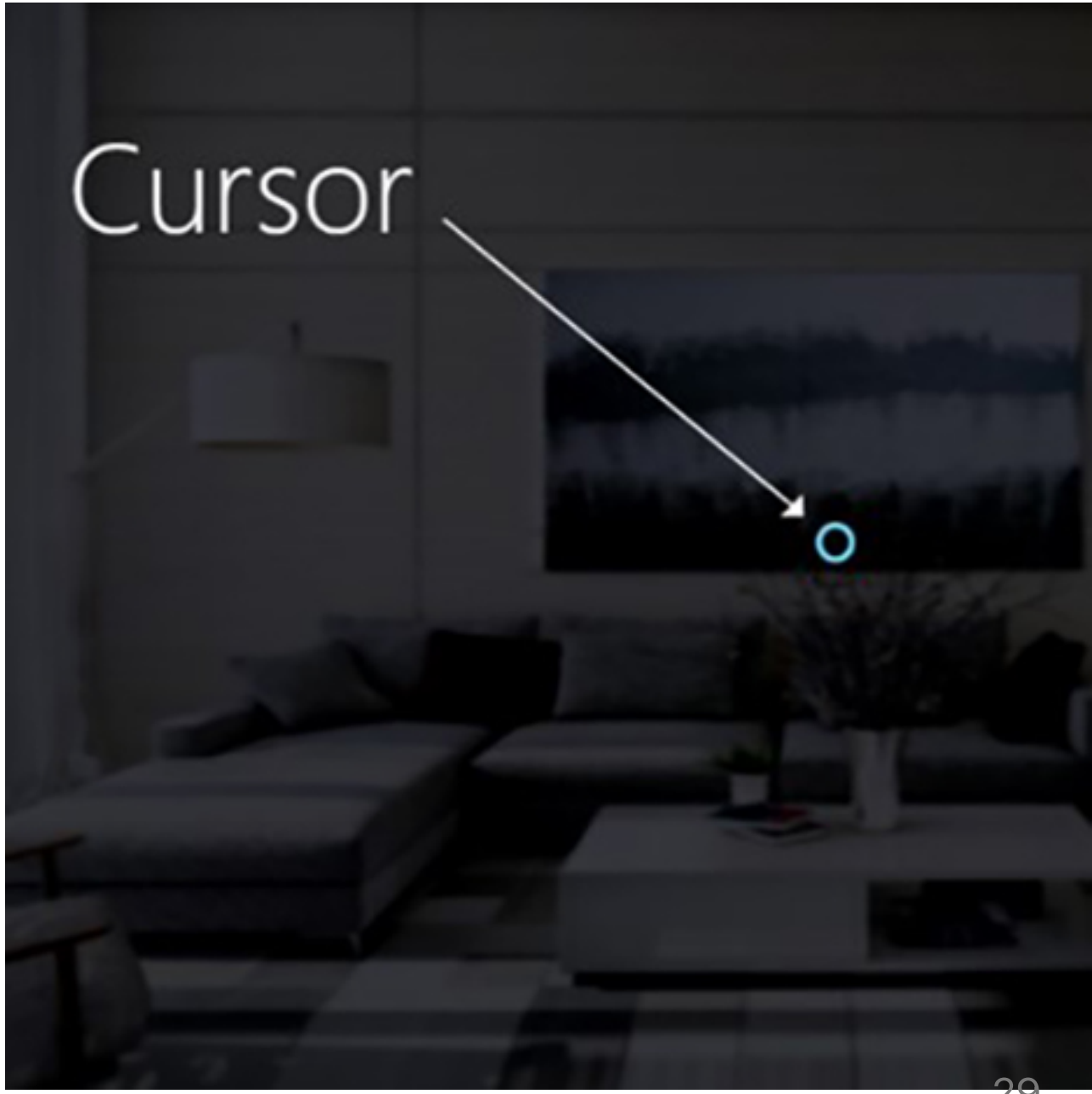
Gaze and dwell is a type of interaction where

- a user looks at an object (targeting), then
- keep looking at the target to select

XR Gaze-based Interaction

Head-gaze and commit is a special type of interaction where a

- use target an object with their head
- perform a secondary input, such as using hand or voice



Unity Core Concepts

Unity Scene

Represents **GameObjects** in a three-dimensional space.

Unity Camera

A component that creates an image of a particular viewpoint in your scene.

It is the player's eye in a scene. Unity supports both perspective (realistic) and orthographic (fixed size) cameras.

GameObject

GameObject class represents any object that can exist in a **Scene.**

It provides methods to work with the scene's object in code.

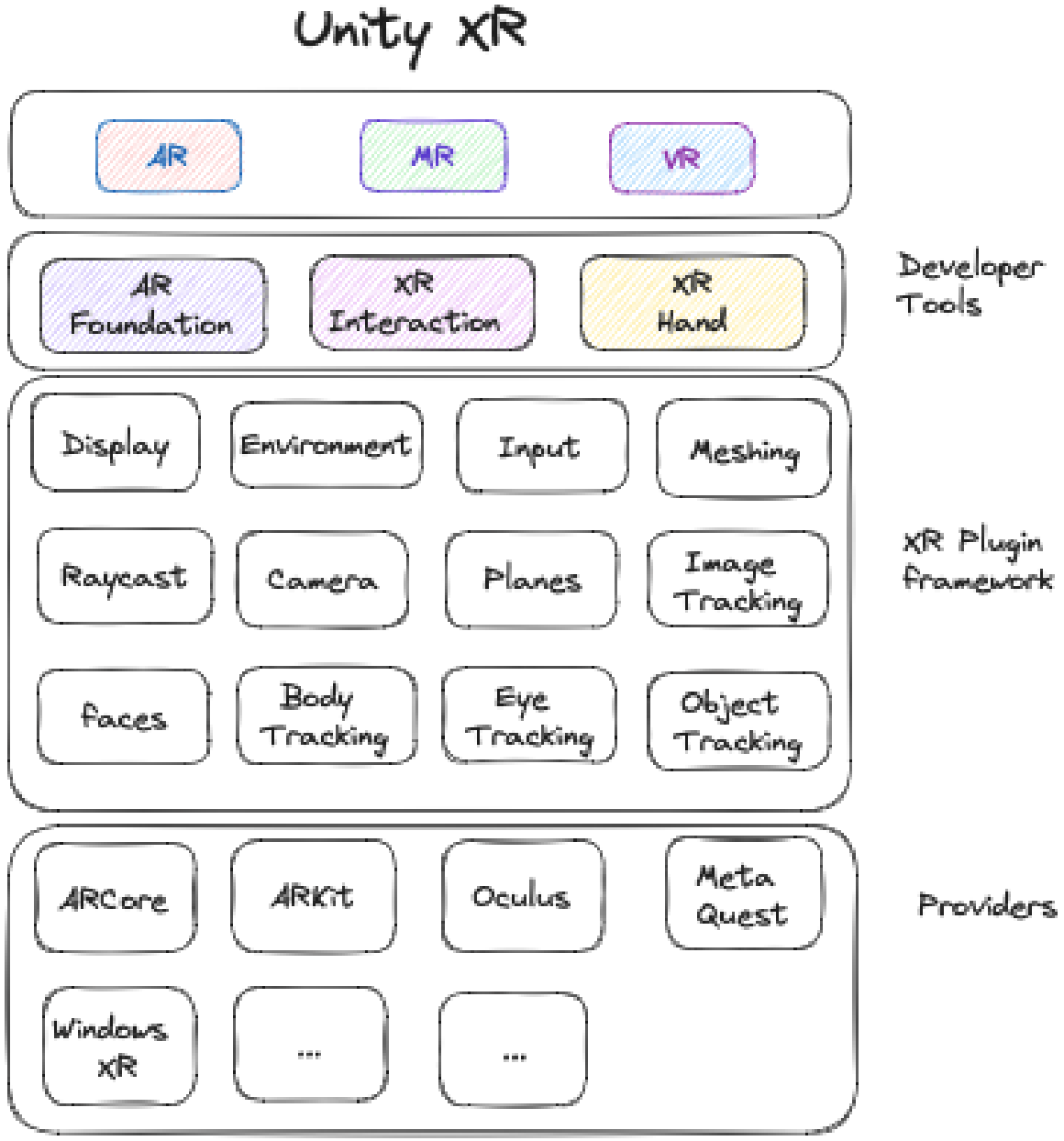
Methods include:

- finding
- making connections and sending messages between GameObjects
- adding and removing components attached to GameObjects
- setting values to GameObjects' status

Component

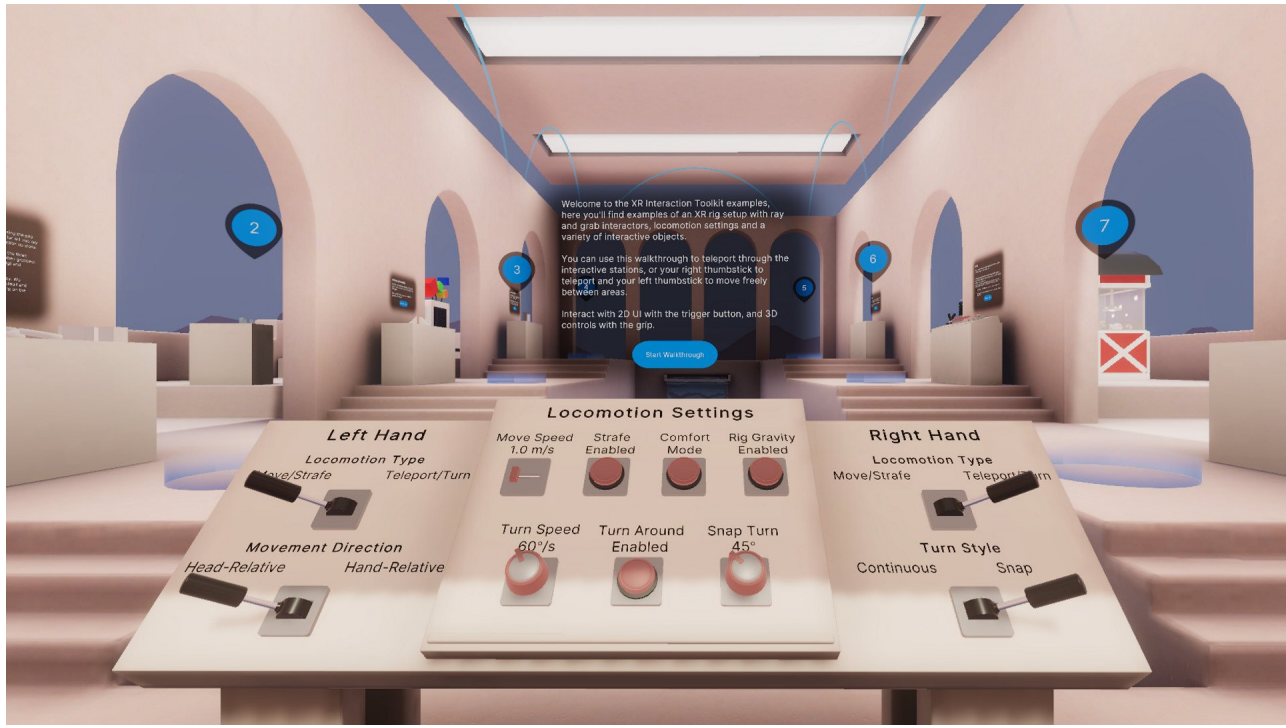
Components are the building blocks of Unity. Game objects are nothing but containers of components.

XR Unity Tech Stack



AR Foundation

XR Interaction Toolkit



XR Hand Toolkit

XR Unity Input

Is a package that allows users to interact with an XR application using a device, touch, or gestures.

Unity offers a wide range of inputs

- Keyboards and mice
- Joysticks
- Controllers
- Touch screens
- Movement-sensing capabilities of mobile devices, such as accelerometers or gyroscopes

XR Unity Raycast

Raycast is one of the main techniques used for spatial interaction and collision detection in XR. The term "raycast" refers to the process of projecting an imaginary ray (or line)

- From a starting point
- Into a specific direction in the 3D space

Raycast enables us to determine what objects or surfaces the ray intersects