Version: 20 Mar 2024

Quick Start

This section includes information on how to get started using Hand Tracking via Unity's XR Hands Package and the OpenXR Hand Interaction Profile. See the Unity Manual for more information about the XRHands Package and OpenXR Input.

Prerequisites

- Unity Editor (2022.3 LTS)
- OpenXR Plugin (Installed through the Package Manager)
- Magic Leap 2 Package (Installed through the Package Manager)
- Magic Leap 2 Unity SDK (Downloaded via the Magic Leap Hub)

> **CAUTION**
>
> This feature requires the `HAND_TRACKING` permission to be **enabled** in your project's Permissions Settings (**Edit > Project Settings > Magic Leap > Permissions**).

Enable the Hand Interaction Profile

The Hand Interaction Profile in Unity's OpenXR Plugin allows your XR application to leverage the capabilities of hand tracking devices. This profile exposes a specific device layout called `<HandInteraction>` within the OpenXR runtime, enabling interaction with gestures and action poses instead of traditional controller buttons.

This section includes instruction on how to enable the Hand Interaction Profile in your Unity project.

1. Navigate to **Project Settings > XR Plug-in Management**.
2. Verify that **OpenXR** and **Magic Leap feature group** is selected inside the **Plug-in Providers**
3. Select **OpenXR** from the sidebar
4. Click the **+** button inside the **Enabled Interaction Profiles** panel.

5. Select **Hand Interaction Profile** from the list of available profiles.

Install XR Hands Package

1. Open the Package Manager window (**Window > Package Manager**).
2. In the search bar, type "XR Hands" and select the **com.unity.xr.hands** package.
3. Click **Install**. Wait for the installation to complete.

Enable Hand Tracking Subsystem

To use the XRHands package, you must enable the XRHandSubsystem. The subsystem allows scripts to easily subscribe to updates, using XRHandSubsystem.updatedHands

1. Open the OpenXR Project Settings (**Edit > Project Settings > XR Plug-in Management > OpenXR**)
2. Enable Hand Tracking Subsystem under **OpenXR Feature Groups**.

Importing XR Hands Sample

Once the XR Hands package is installed, find it in the Package Manager. Expand the package details and locate the Samples section. Import the XR Hands sample by clicking on the 'Import' button next to it.

Scene Setup

1. Go to **File > New Scene** to create a new scene or open an existing scene where you want to use XR Hands.
2. In the Hierarchy window, right-click and select `XR/XR Origin (AR)` if it is not already in the scene.
3. Locate the **Camera Offset** GameObject under the **XR Origin** in the Hierarchy.
4. Add the **Left Hand Tracking** and **Right Hand Tracking** prefabs from `Assets/Samples/XRHands/<VERSION>/HandsVisualizer/Prefabs/` as children of the **Camera Offset**.
5. Adjust the position and settings of the XR Hands to fit your scene setup.

Using the Hand Interaction Input Profile Actions

This section provides a brief overview on how to detect Magic Leap 2 hand tracking input Unity's `InputAction` class. For a more in depth guide, see the Unity Input System and OpenXR Input guides.

This example initializes actions for the Left Hand Pointer Pose and the Left Hand Grasp Valueand logs a message while the trigger is pressed.

> **INFO**
>
> See the Input Bindings documentation for a complete list of Magic Leap 2 Controller bindings.

```
using UnityEngine;
using UnityEngine.InputSystem;

/// <summary>
/// This script shows how to create an Input Action at runtime.
/// </summary>
public class InputActionTest : MonoBehaviour
{
    //The input action that will log it's Vector3 value every frame.
    [SerializeField]
    private InputAction positionInputAction =
        new InputAction(binding:"<HandInteraction>
{LeftHand}/pointerPosition", expectedControlType: "Vector3");

    //The input action that will log the grasp value.
    [SerializeField]
    private InputAction graspValueInputAction =
        new InputAction(binding: "<HandInteraction>
{LeftHand}/graspValue", expectedControlType: "Axis");

    private void Start()
    {
        positionInputAction.Enable();

        graspValueInputAction.Enable();
    }

    private void Update()
    {
        // Print pointer position to log.
        Debug.Log($"Left Hand Pointer Position:
{positionInputAction.ReadValue<Vector3>()}");

```

```
        // Print the left hand grasp value to log.
        Debug.Log($"Left Hand Grasp Value:
{positionInputAction.ReadValue<float>()}");
    }

    private void OnDestroy()
    {
        triggerInputAction.Dispose();
        positionInputAction.Dispose();
    }
}
```

See Also:

- Unity OpenXR - Input Manual
- Unity Input System - Actions
- Unity Input System - Class InputAction

Tags: Unity Example Input