WRIST AND HAND TRACKING

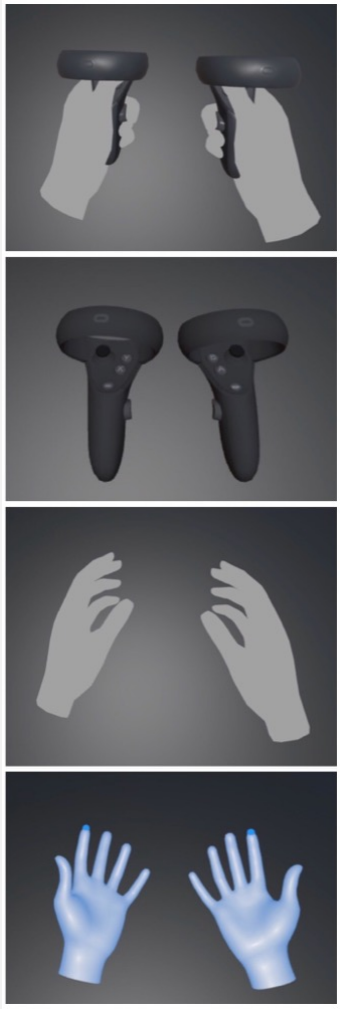
Corso Realtà Virtuale 2022/2023

eleonora.chitti@unimi.it

GESTURES AND INTERACTION IN VR

There are different kinds of interaction modes in VR, as stated by [2]

- Use controllers and
  - Show the hands grabbing the controllers in the VR world
  - Show the controllers in the VR world
  - Show hands instead of the controllers in the VR world
- Use Hand tracking and
  - Show hands tracked in the VR world
  - Recognize gestures, as the swipe gesture, or drag up/down/left/right …



Four possible interaction modes, image from [2]

TECHNOLOGIES FOR HAND TRACKING

Any contact with the real world such as sensors or controllers can break the immersion in the Virtual World [1]

Therefore, technologies as the following have been commercially deployed:

- Oculus - Quest hand tracking
  https://support.oculus.com/2720524538265875/
- Ultraleap - Leap Motion Hand tracking (also with the VR Developer Mount)
  https://www.ultraleap.com/tracking/
- Haptics sensors, for example the Ultraleap - Stratos
  https://www.ultraleap.com/haptics/

LEAP MOTION CONTROLLER

Image from https://www.ultraleap.com/tracking/

The Leap Motion Controller is a free-hand interaction controller, to control input with hand movements, and it can be plugged via USB cable connection.

The sensors work with infrared light, and the device has a field of view of about 140x120° ; the range is approximately from 0.03 up to 0.6 - 0.8 meters above the device.

The detection and tracking work best when the controller has a clear view of a hand's silhouette. However, the Leap Motion software combines its sensor data with an internal model, with bones and joints, of the human hand to cope with challenging tracking conditions and "accurately predict the position of a finger or thumb, even if it's hidden from view".

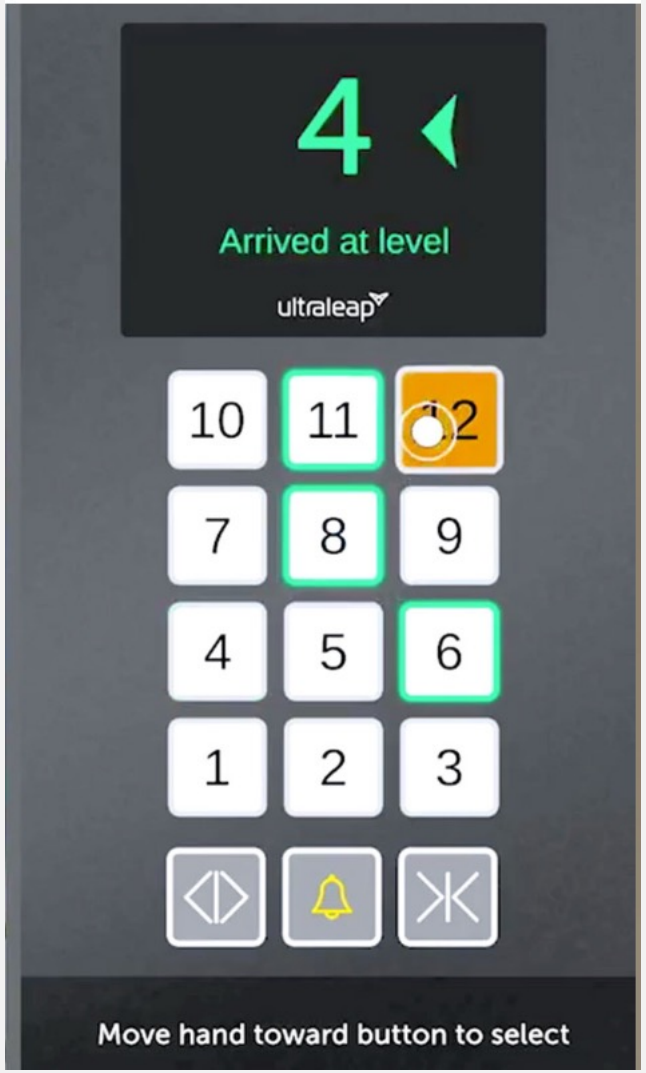
https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf

TOUCH-FREE WITH LEAP MOTION CONTROLLER

TOUCH FREE (LEAP MOTION) SDK

TouchFree is a software application that detects a user's hand in mid-air and converts it to an on-screen cursor, to allow touchscreen-style interactions.

It supports both modalities hover or tap to click.



The TouchFree can be mounted in 3 modes:

- On top of the screen pointing in the screen direction
- On top of the screen pointing in user's direction
- On the bottom of the screen

https://gallery.leapmotion.com/touchfree/

TOUCH FREE (LEAP MOTION) SDK

To install TouchFree you can follow the manual, before importing any development package you should install TouchFree SDK:

https://docs.ultraleap.com/touchfree-user-manual/setup.html

The Ultraleap - TouchFree offers tools for Unity and for Web Apps.

You can download the TouchFree unitypackage here:

https://developer.leapmotion.com/touchfree-tooling-unity

Documentation

Unity:

https://docs.ultraleap.com/touchfree-user-manual/tooling-for-unity.html

Web (HTML):

https://docs.ultraleap.com/touchfree-user-manual/tooling-for-web.html

HAND TRACKING WITH LEAP MOTION CONTROLLER

LEAP MOTION SDK

To work with Leap Motion you need to install the SDK on the device, currently only devices with Windows OS are supported with the latest version of the SDK 4.1.0 (V4 Orion)

MacOS and Linux were supported until SDK 2.3.1 (V2)

https://developer.leapmotion.com/tracking-software-download/

The Leap Motion can be used in two modes:

- the VR Headset setup
  https://developer.leapmotion.com/vr-headset-leap-motion-controller/

- the Desktop/Laptop setup
  https://developer.leapmotion.com/desktop-leap-motion-controller/

LEAP MOTION SDK

A sum up of software versions' differences https://developer.leapmotion.com/documentation/

- **V2** the last one supporting MacOS and Linux, it supported tool tracking or touchscreen-style gestures, and Unity, Unreal, C++, C#, Objective-C, Java, Python, and JavaScript languages.

- **V3** (Orion beta – Orion first version) first version optimized for the VR setup, it preserved many of the legacy APIs (including same as V2 languages support), but not tool tracking.

- **V4** (Orion) second generation of Orion, optimized for VR, it supports - latest stable version
- **V5** (Gemini)
  - C# with Unity
    https://developer.leapmotion.com/unity
  - Unreal
    https://developer.leapmotion.com/unreal
  - LeapC api (a C-style API for accessing tracking data from the Leap Motion service)
    https://developer.leapmotion.com/documentation/v4/index.html

SDK AND UNITY MODULE SETUP

Here the steps to install the Leap Motion SDK and Unity Package (Desktop mode):

Install the latest SDK, to have the Leap Motion Software running. From the top icon of the sw now you can:

1. **Stop Tracking** and **Resume Tracking**
2. Run the **Leap Motion Visualizer** to check if the device is working properly.
3. Run the **Leap Motion Control Panel**
   it will show you the hands tracked, and it permits the troubleshooting with the recalibration of the device

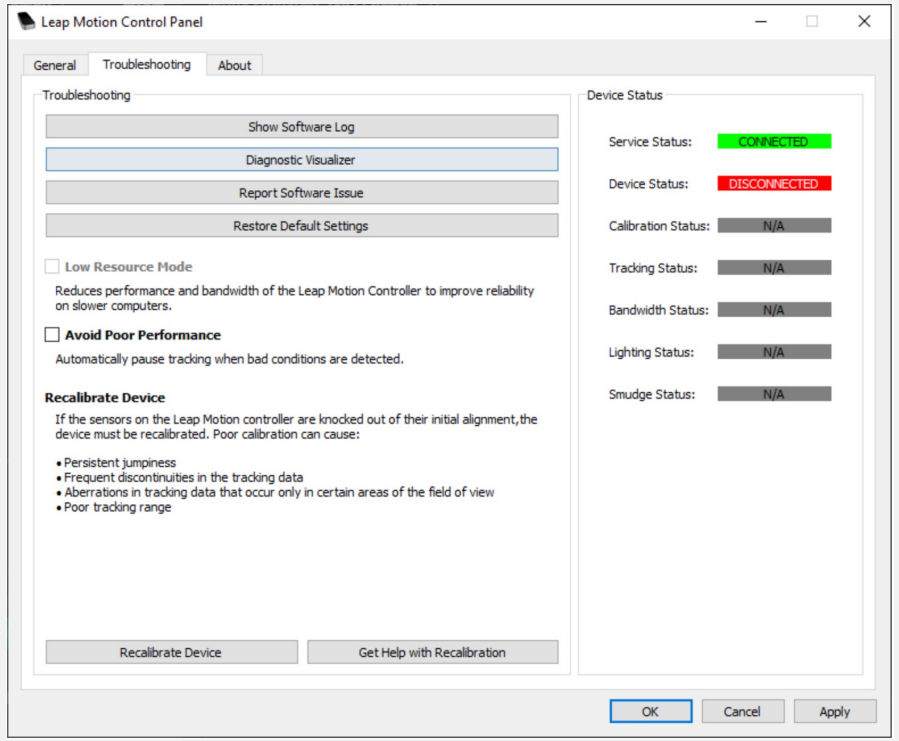


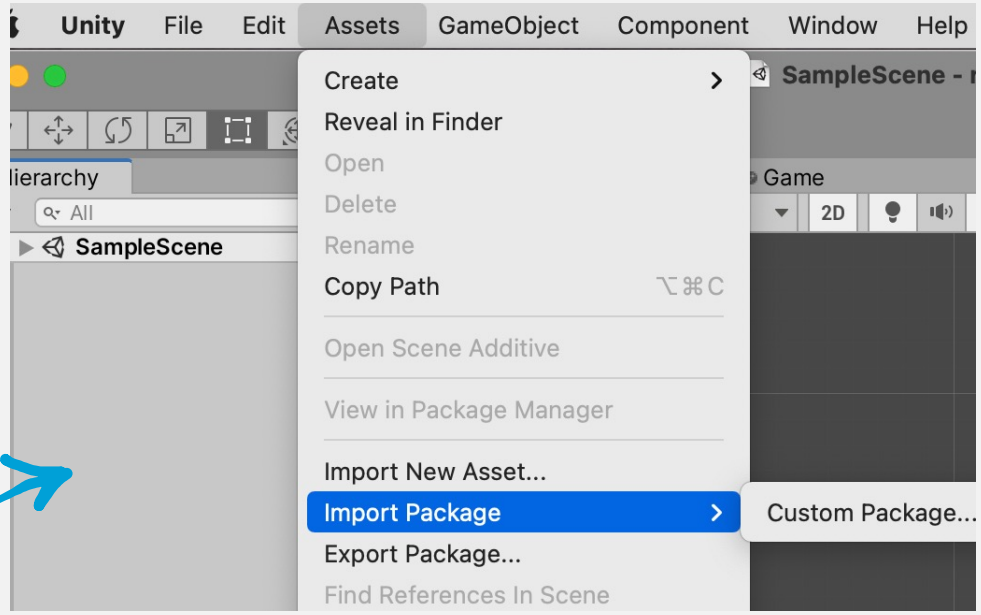
Image from https://developer.leapmotion.com/desktop-leap-motion-controller/

SDK AND UNITY MODULE SETUP

3. Download the Leap Motion Unity Core Assets package, that can be imported into Unity3D as a *Custom Package*.
   https://developer.leapmotion.com/unity/
   or
   https://github.com/ultraleap/UnityPlugin/releases/

4. Open the Unity Project in which you want to exploit the Leap or create a new Unity Project

5. Click on Assets > Import Package > Custom Package and select the Leap Motion unitypackage just downloaded



6. Now in the Unity's Project Panel will appear LeapMotion folder, that contains the Prefabs, the Scripts, and the Demo Scenes (in addition the Plugins folder and the LeapC folder with all leap motion API bindings will appear)

UNITY MODULES

Unity Core Assets package can be downloaded here: https://developer.leapmotion.com/unity/

The Unity Core Package includes:

- Interaction Module (Interaction Engine)
  https://leapmotion.github.io/UnityModules/interaction-engine.html
  That allows users to work with your XR application by interacting with physical or pseudo-physical objects, as balls or dices.
- Hands Module https://leapmotion.github.io/UnityModules/hands-module.html
  That allows to use already available hands assets in the Unity Leap package, but it also allows to auto-rig a wide hand types made with FBX extension.

UNITY3D LEAP MOTION FEATURES: INTERACTION

Interaction (Interaction Engine):

This engine will track the collision and interaction with other GameObject (GO) in the scene as picking them or touching them.

The Leap Motion virtual hands are rigged and have also rigidbodies and colliders to manage the physics interaction in the virtual world.

For example, a demo project included in the elder V2 SDK show how to pick up a flower's petal (image in the next slide).

To allow the interaction the Leap Objects will have the *InteractionManager* and the *InteractionController* components.

- The *InteractionManager* manages the physics of the interaction from FixedUpdate data.
- *The InteractionController* manages the real interaction with the other Game Object (GO).

https://leapmotion.github.io/UnityModules/interaction-engine.html

UNITY3D LEAP MOTION FEATURES: INTERACTION)

Interaction (Interaction Engine):

The other Game Object in the scene to be interactable with the Leap Object should have a

- *InteractionBehaviour* component to manage the type of possible interaction of hands on that GO (for example the Petal GO should have the pick-up behavior)
- A Rigid Body and a Collider, that are automatically added when the *InteractionBehaviour* component is added to that GO.

Image from https://blog.leapmotion.com/category/code/foss/page/5/

UNITY3D LEAP MOTION FEATURES: INTERACTION

Interaction (Interaction Engine):

In the Unity Scene you should have the following GO:

Image from https://leapmotion.github.io/UnityModules/interaction-engine.html

More info on the interaction behaviours and physics available here:
https://leapmotion.github.io/UnityModules/interaction-engine.html#ie-working-with-physx

UNITY3D LEAP MOTION FEATURES: HANDS

Hands Module:

In the Leap Motion Unity Package are available already rigged and Leap Motion compatible hands.

However, if you can use a custom FBX model using the **LeapHandsAutorig** Monobehaviour Script that runs in the Unity Editor.

Using this feature the custom hands model is automatically auto-rigged to make it compatible with the Leap Motion. The FBX can be defined as humanoid or not, the script acts differently in case of not humanoid, however both cases are supported.

Doc about Auto-rigging is available here:
https://leapmotion.github.io/UnityModules/hands-module.html

From that link a good point: to remember: "Beware the Uncanny Valley: Hyper-realism isn't always the best approach in VR. Users almost always respond better to stylized or cartoony hands"

ONE EXERCISE (FIRST PART)

1. Import the leap motion "*Tracking Examples*" UnityPackage (you can find it in the zip folder from https://developer.leapmotion.com/unity/)
   (In an error appears on the Console: go to Windows -> Package Manager -> Unity Registry search for Input Manager, then install it + Import the *Tracking.unitypackage* before importing the Tracking examples)

2. Import the fruit UnityPackage (available on laboratory github)

3. Open Food Scene

4. In the scene add the Prefab *Interaction Manager* that you can find in
   Assets ->Third Party -> Ultraleap -> Tracking -> Interaction Engine -> Runtime folder

5. In the scene add the Prefab *Capsule Hands.prefabs* that you can find in
   Assets ->Third Party -> Ultraleap -> Tracking -> Core-> Runtime folder -> Prefabs -> Hands

6. In the scene adds the Prefab *Leap Service Provider (Desktop)* that you can find in
   Assets -> Third Party -> Ultraleap -> Tracking -> Core-> Runtime folder

ONE EXERCISE (FIRST PART)

8. In the scene add the Pumpkin Prefab
9. Select the Pumpkin, then in the inspector add a *Mesh Collider* and check "convex"
10. Always in the Pumpkin inspector add the *Interaction Behaviour (Script)*
11. Now run in the editor and try to grab the pumpkin with the hand
12. Now save the updated Pumpkin prefab in the Prefabs folder

ONE EXERCISE (SECOND PART - OPTIONAL)

1. In the Scene click on the object Cube UI Button Base (child of the object Cube)

2. Add the *Cube UI Button* as child to the Cube UI Button Base that you can find in Assets ->Third Party -> Ultraleap -> Tracking -> Examples -> Interaction Engine Examples ->Common Example Assets -> Prefabs

3. Select the *Cube UI Button*, then in the inspector in the *Interaction Button (Script)* associate the public variable "Manager" with the Interaction Manager in the scene

4. Select the *TestScript* object in the scene, and in the inspector in the public variable *Items* add the Pumpkin prefab in the Prefabs folder

ONE EXERCISE (SECOND PART - OPTIONAL)

5. Click again on the *Cube UI Button*, then in the inspector in the *Interaction Button (Script)*

   - In the *PerControllerPrimaryHoverEnd()* click on +

   - Now add here the TextScript and select the function

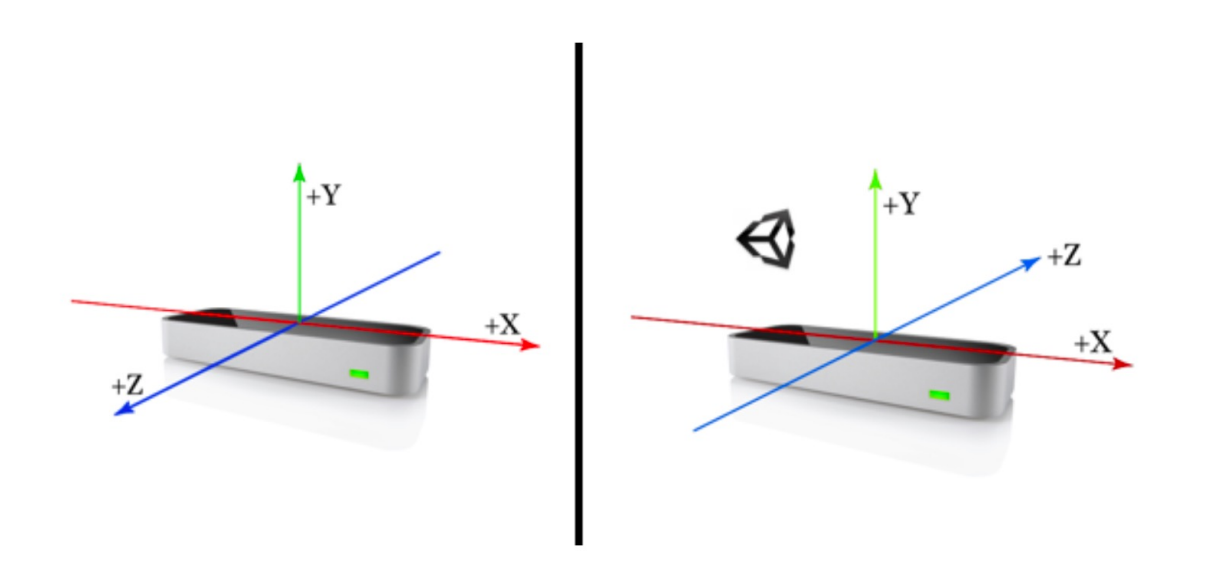
   - DeleteAndReinstantiate.DeleteAndInstantiate()

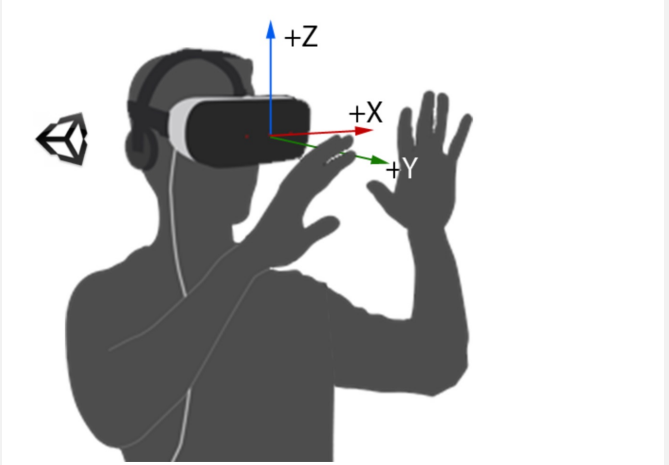
6. Now run in the editor and push the button

MORE ABOUT LEAP MOTION: LEGACY DOCUMENTATION

In the developer-archive website you can find Legacy SDK, Unity Modules and an accurate documentation about V2 and V3 on Leap Motion features as the Leap Motion Coordinate System:
https://developer-archive.leapmotion.com/documentation/v2/unity/unity/Unity_Overview.html

The Leap Motion has a Right-Hand coordinate system; however, the Unity 3D environment uses the Left-Hand. Unity also uses a default unit of meters, whereas the Leap Motion API uses millimeters.
The Unity Leap Motion plugin scripts internally transforms the tracking data to use the left-handed coordinate system and scales distance values to meters.





The Unity coordinate system superimposed on the Leap Motion device in its HMD orientation.

MORE ABOUT LEAP MOTION: UNITY FEATURE OPENXR

In December 2020 Unity announced the 2020.2 preview feature of OpenXR to simplify AR/VR development with Unity targeting a wide range of devices, including Leap Motion.

The OpenXR plug-in is implemented as a part of the XR plug-in framework https://docs.unity3d.com/Manual/XRPluginArchitecture.html

Here the Unity announce and OpenXR demo explanation:
https://forum.unity.com/threads/unity-support-for-openxr-in-preview.1023613/

Here the Leap Motion demo explanation with OpenXR:

https://docs.ultraleap.com/ultralab/openxr-hand-tracking-in-unity/

MORE ABOUT LEAP MOTION: MIXED REALITY TOOLKIT (MRTK)

The Mixed Reality Toolkit (MRTK) is a Microsoft-driven project that provides a set of components and features, to support cross-platform mixed reality development in Unity.

The Leap Motion is compatible with MRTK.

MRTK docs page:

- Elder: https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/WelcomeToMRTK.html
- Latest: https://docs.microsoft.com/it-it/windows/mixed-reality/mrtk-unity/

Microsoft docs page for setup Leap Motion with MRTK:

- Elder: https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/CrossPlatform/LeapMotionMRTK.html
- Latest: https://docs.microsoft.com/it-it/windows/mixed-reality/mrtk-unity/features/cross-platform/leap-motion-mrtk

Leap Motion website about MRTK:

https://developer.leapmotion.com/mrtk-unity/

REFERENCES

[1] R. McGloin, K. Farrar, and M. Krcmar, "Video games, immersion, and cognitive aggression: does the controller matter?" Media psychology, vol. 16, no. 1, pp. 65–87, 2013

[2] Jan-Niklas Voigt-Antons and Tanja Kojić and Danish Ali and Sebastian Möller, "Influence of Hand Tracking as a way of Interaction in Virtual Reality on User Experience", 2020, 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX), arXiv: https://arxiv.org/abs/2004.12642