VIVE Tracker (3.0) Developer Guidelines

Ver. 1.0

Release Notes

Version no.	Release date	Content
1.0	2021.01.18	Initial public version for VIVE Tracker (3.0)

Table of Contents

Introduction

This document describes the development guidelines for VR accessory makers and content developers. It contains information on how to use the VIVE Tracker (3.0) to enable positional tracking and transmission of specific data (with or without the HTC VIVE VR system).

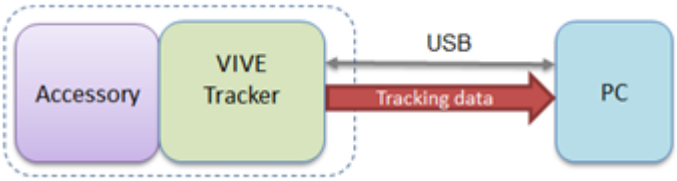
VIVE Tracker (3.0) can pair with HTC's wireless dongle or use its USB interface to transfer tracking data to a PC. An accessory attached to VIVE Tracker (3.0) can:

- Simulate buttons of the VIVE Controller through the underlying Pogo pin port.
- Send specific data to a PC through the USB interface of VIVE Tracker (3.0).

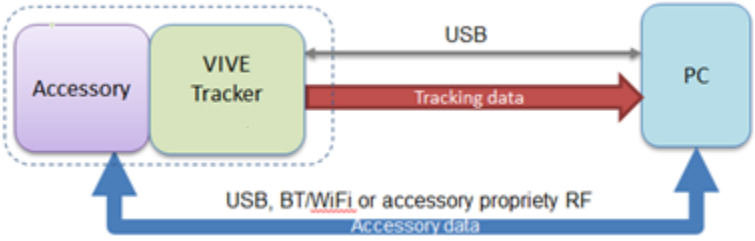
Use cases

There are five use cases supported by VIVE Tracker (3.0).

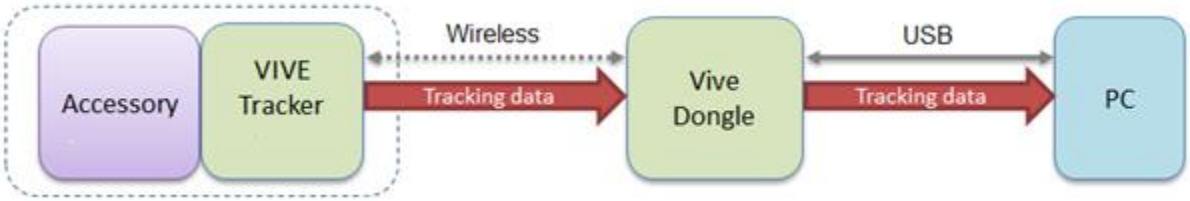
Use Case 1: Track passive objects through USB interface in VR. In this case, the dongle is not used. VIVE Tracker (3.0) is connected to the PC through USB to directly transfer tracking data.



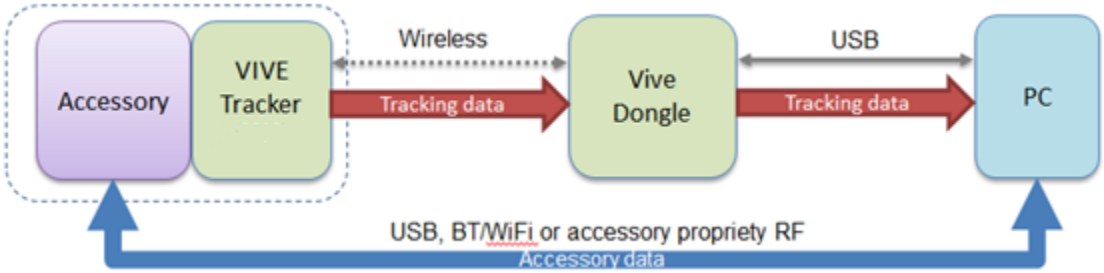
Use Case 2: Track passive objects through USB interface in VR, with the accessory passing data to a PC through USB, BT/Wi-Fi or propriety RF. This is similar to Use Case 1 but the accessory directly transfers the tracking data to a PC for a specific purpose based on its design.



Use Case 3: Track moving objects by wireless interface in VR. In this case, the dongle is used to transfer tracking data from the VIVE Tracker (3.0) to a PC.



Use Case 4: Track moving objects using a wireless interface in VR, with the accessory passing data to a PC through USB, BT/Wi-Fi or propriety RF. This is similar to Use Case 3 but the accessory directly transfer the tracking data to/from a PC for a specific purpose based on its design.

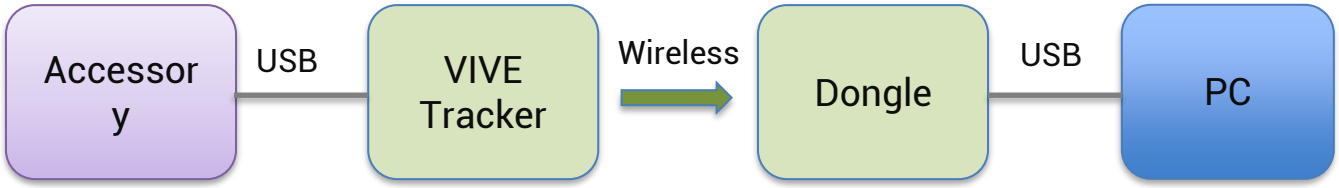


Use Case 5: Track moving objects using a wireless interface in VR, with the accessory simulating buttons of the VIVE Controller or passing data to a PC through the VIVE Tracker (3.0). This is similar to Use Case 3 but the accessory connects with the VIVE Tracker (3.0) to transfer a button event to a PC through the Pogo pins or USB interface.

Hardware requirements

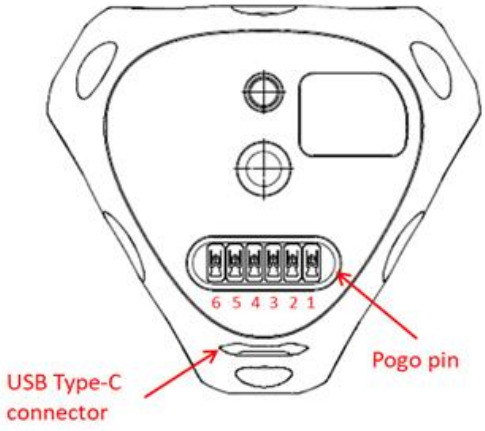
This section describes hardware requirements for accessories used with the VIVE Tracker (3.0) in order to enable positional tracking and input of specific data for the HTC VIVE VR system.

A compatible accessory can be attached to the VIVE Tracker (3.0) to send specific data to a PC through the USB interface of the VIVE Tracker (3.0). The VIVE Tracker (3.0) needs to be paired with the dongle first to be able to transfer an event to a PC. The figure below describes the conceptual architecture.



Interface

USB 2.0 full speed (client) from USB Type-C connector.



GPIO Pin Absolute Maximum Rating

Symbol	Parameter	Min	Max	Unit
V_I	Input voltage	- 0.3	3.6	V
V_{ESD}	Electrostatic discharge voltage , Human Body Model	--	4000	V

GPIO Pin Electrical Characteristics (Supply voltage VDD = 3.3 V)

Symbol	Parameter	Min	Typ	Max	Unit
V_{OH}	High-level output voltage	VDD - 0.4	--	--	V
V_{OL}	Low-level output voltage	--	--	0.4	V
V_{IH}	High-level input voltage	0.7VDD	--	--	V
V_{IL}	Low-level input voltage	--	--	0.3VDD	V
I_{OH}	High-level output current		--	2	mA
I_{OL}	Low-level output current		--	2	mA
I_{IH}	High-level input current	-1	0.5	1	nA
I_{IL}	Low-level input current	-1	0.5	1	nA

Radio frequency (RF)

To establish a stable wireless connection between the VIVE Tracker (3.0) and the dongle, the OTA performance of VIVE Tracker (3.0) cannot degrade to more than 3dB when an accessory is attached to the VIVE Tracker (3.0).

The following are recommendations for better RF performance.

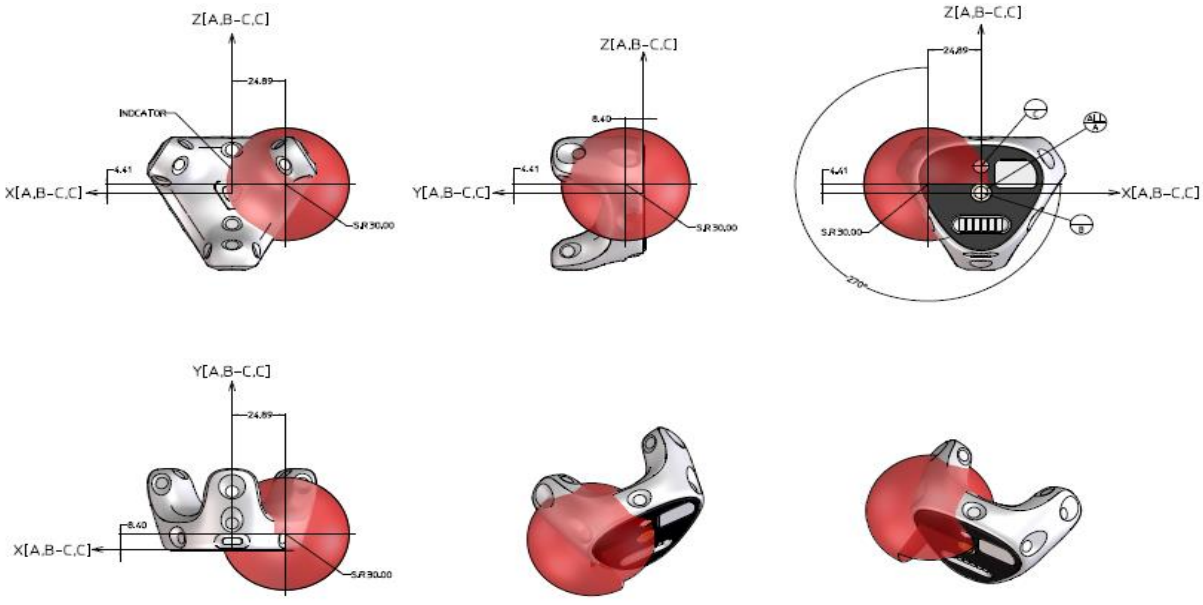


Figure: Illustrates the "keep out" area where only nonmetallic parts of the accessory should be inside (spherical radius = 30 mm with the center being the antenna feed point)

Except for essential parts, such as the 1/4'' screw, electric connection pad (which connects with the Pogo pin), and related circuits of the electric connection pad, metal parts of the accessory should be separated from the antenna by at least 30mm to prevent OTA performance reduction when the accessory is attached to VIVE Tracker (3.0).

Power

USB-C Connector	Voltage requirement	Max. Charging current	Charging time (approx.)
AC	5V+/-5%	700 mA	2 hrs
PC		500 mA	3 hrs
Pogo Pin 3	**Voltage requirement**	**Max. Charging current**	**Charging time (approx.)**
PC	5V+/-5%	500 mA	3 hrs

Note
AC: D+ short to D-
PC: D+/D- communication

Optics

The field of view (FOV) of VIVE Tracker (3.0) is 240 degrees. Avoid obstructing the FOV of VIVE Tracker (3.0) as this will block responses from the tracker sensors.

If the parts of the docking extend beyond the recommended placement cone, additional views will be blocked.

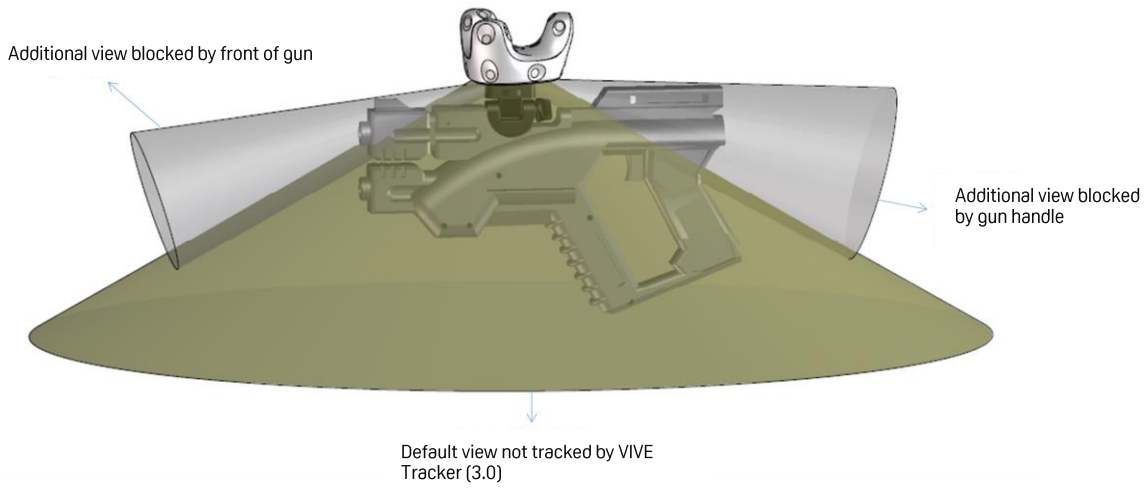


Figure: Part of dock extend beyond recommended placement cone

Docking

Requirements for docking compatibility:

a. The docking design of VIVE Tracker (3.0) follows the ISO standard (ISO 1222:2010). Furthermore, VIVE Tracker (3.0) has some constraints such as the longer screws cannot be screwed all the way in.

b. The user should be able to easily attach and detach VIVE Tracker (3.0) with two hands. One hand holds VIVE Tracker (3.0), and the other hand holds the accessory.

c. The user should not be at risk of physical harm while attaching or detaching VIVE Tracker (3.0).

d. The user should be comfortable while attaching and detaching VIVE Tracker (3.0).

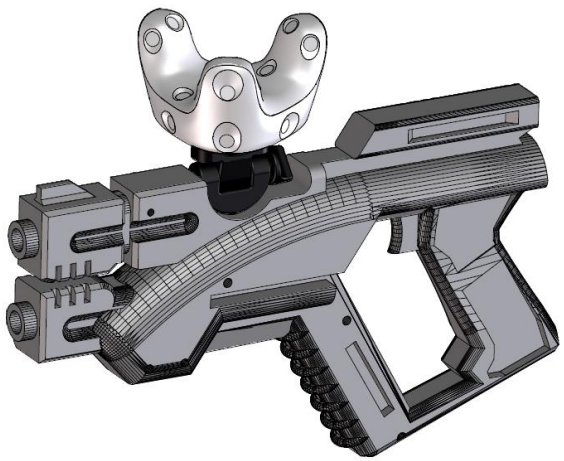
e. The accessory attached with VIVE Tracker (3.0) should be in the shape of a physical object to avoid hitting it while in use.

f. VIVE Tracker (3.0) should not be blocked by the accessory and affect the tracking performance.

g. It is strongly recommended that the accessory uses low reflection materials for its outer skin to avoid reflective interference with the tracking sensors, especially if the accessory needs to be placed inside the FOV of VIVE Tracker (3.0).

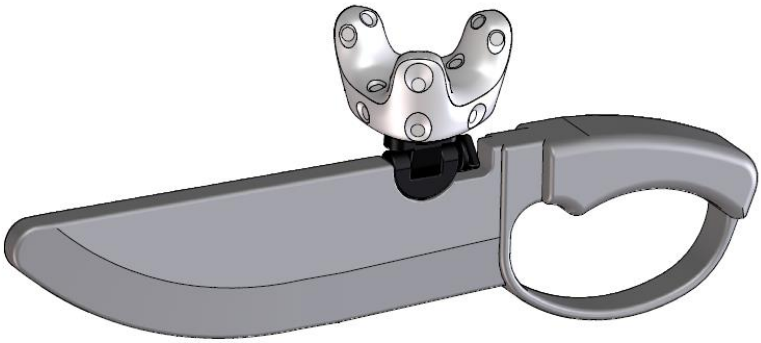
Docking embodiments

Gun



Sword

It is recommended to place the mounting mechanism close to where the sword is held. Also, it is recommended to set the sword length in the VR applications.

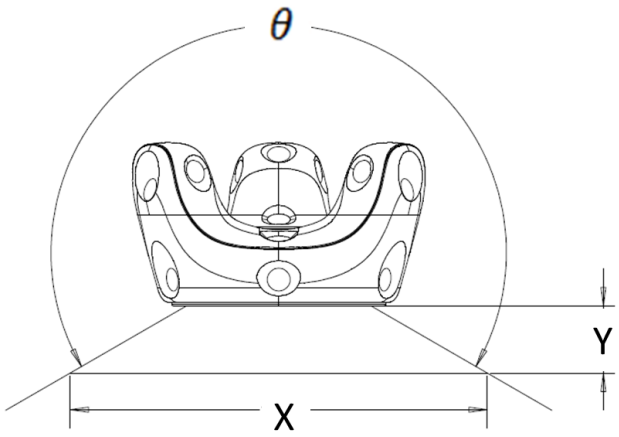


Multi-purpose docking base

Users can attach VIVE Tracker (3.0) to any object/surface that is intended to be tracked.

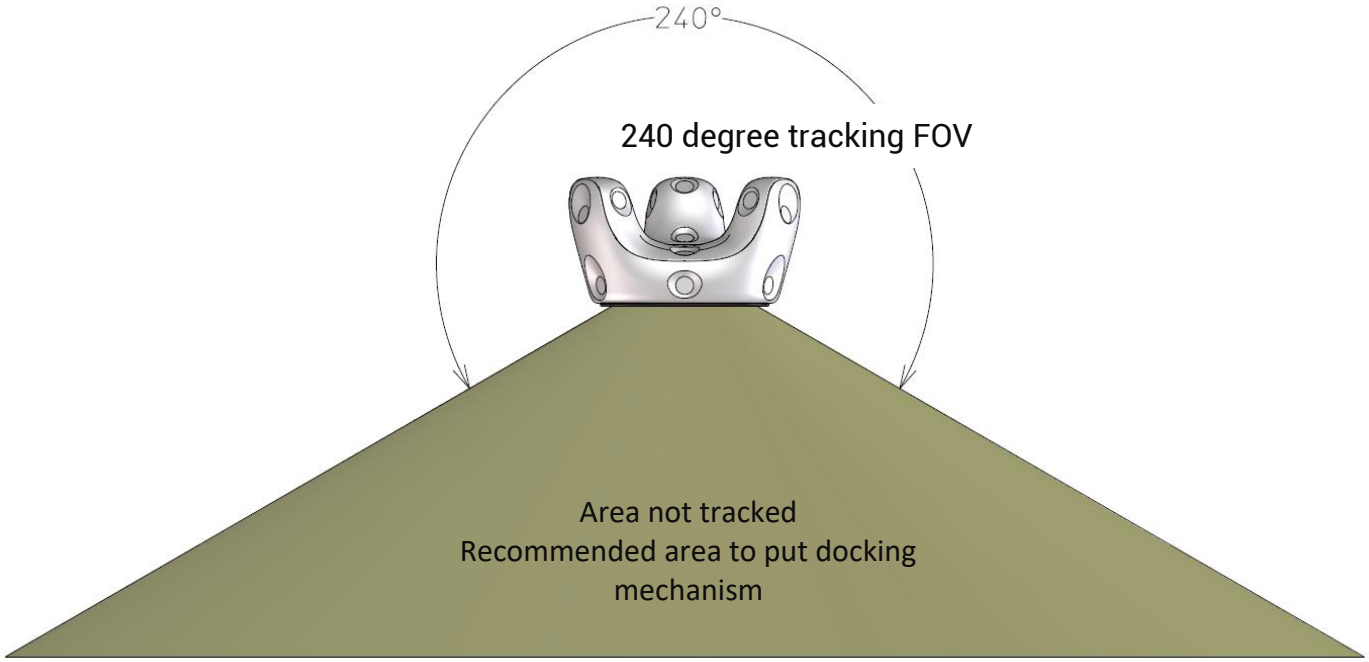
- If the object/surface is smooth and stiff, it is recommended to use a strong adhesive tape for attaching the docking base to the object/surface (ex. 3M VHB tape).
- If the object/surface is rough and soft, it is recommended to use a strap for tightening the docking base to the object/surface.

Improper placement of VIVE Tracker (3.0) may cause a part of the accessory to block the FOV of tracker and therefore affect the tracking performance. The mounting distance between tracking FOV and the related accessory size is shown below:



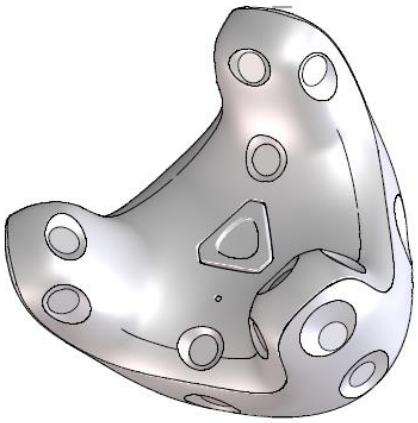
$$\theta = 180° + 2 \times tan^{-1}\frac{2Y}{X}$$

θ : Tracking FOV
X : Docking surface width of accessory
Y : Distance between TRACKER and accessory docking surface

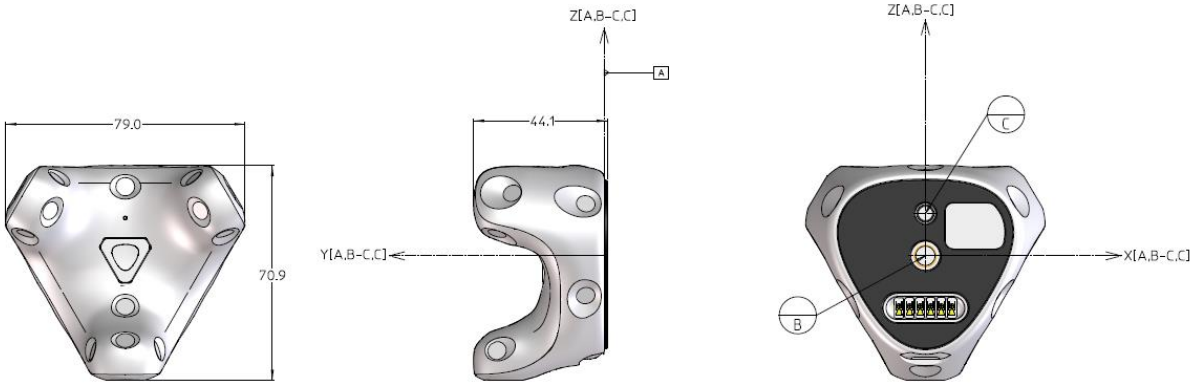


Mechanical considerations

This section describes the mechanical considerations for developers to build various accessories that are compatible to fit or mount with the VIVE Tracker (3.0).



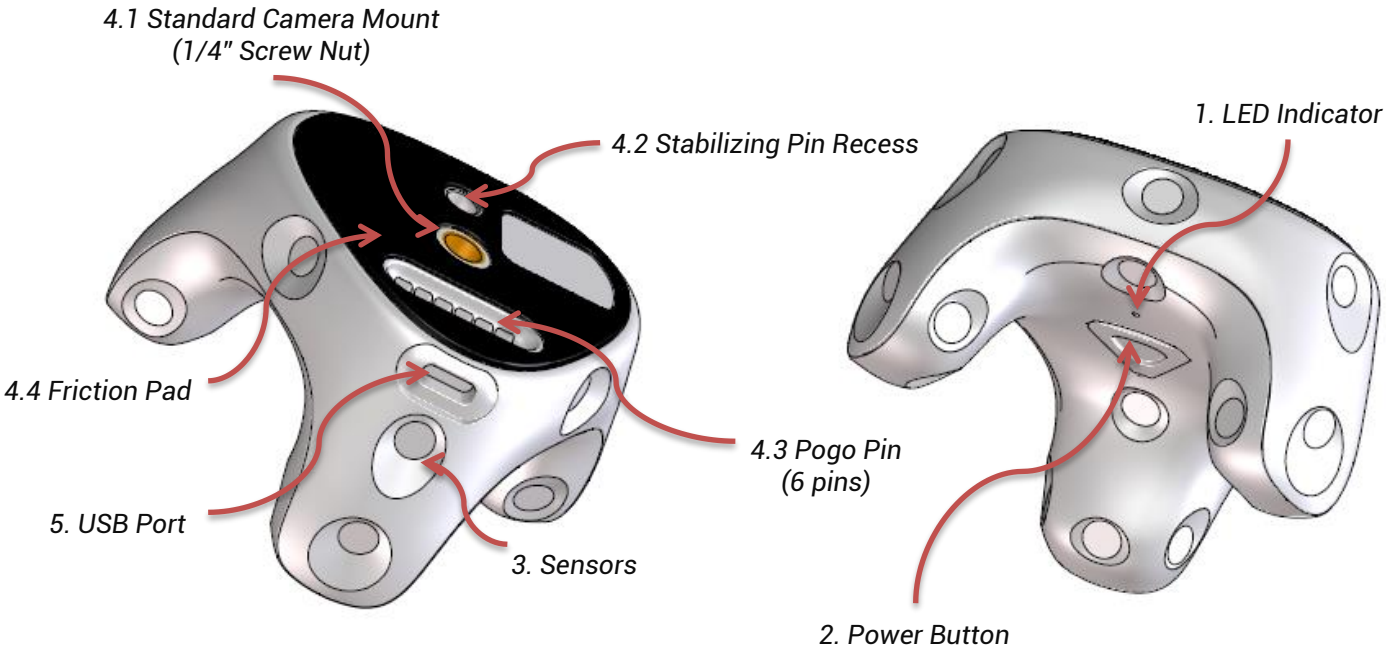
Dimension and weight



Different angles of VIVE Tracker (3.0)

Dimension: 79.0mm (L) x 70.9mm (W) x 44.1mm (H)
Weight: 75g

Main Features



1. **LED Indicator**: Shows the status of VIVE Tracker (3.0).

2. **Power Button**: Used for powering on/off, BLE pairing, etc.

3. **Sensor**: Receives signals from the base stations. The VR system uses the received signals for computing the current location of VIVE Tracker (3.0). Accessories should minimize surface reflection (e.g. avoid white) since it may cause faulty signal and affect performance. Anti-reflective painting is recommended.

4. **Docking Mechanism**: Standard camera tripod docking method is used which is comprised of:
   4.1 1/4" Screw nut to fasten the accessory.
   4.2 Stabilizing pin recess for constraining the tracker from rotating.
   4.3 Pogo pin port (spring contact-type) for optional electrical connection to the accessory.
   4.4 Friction pad to provide a surface with friction between the accessory and VIVE Tracker (3.0)

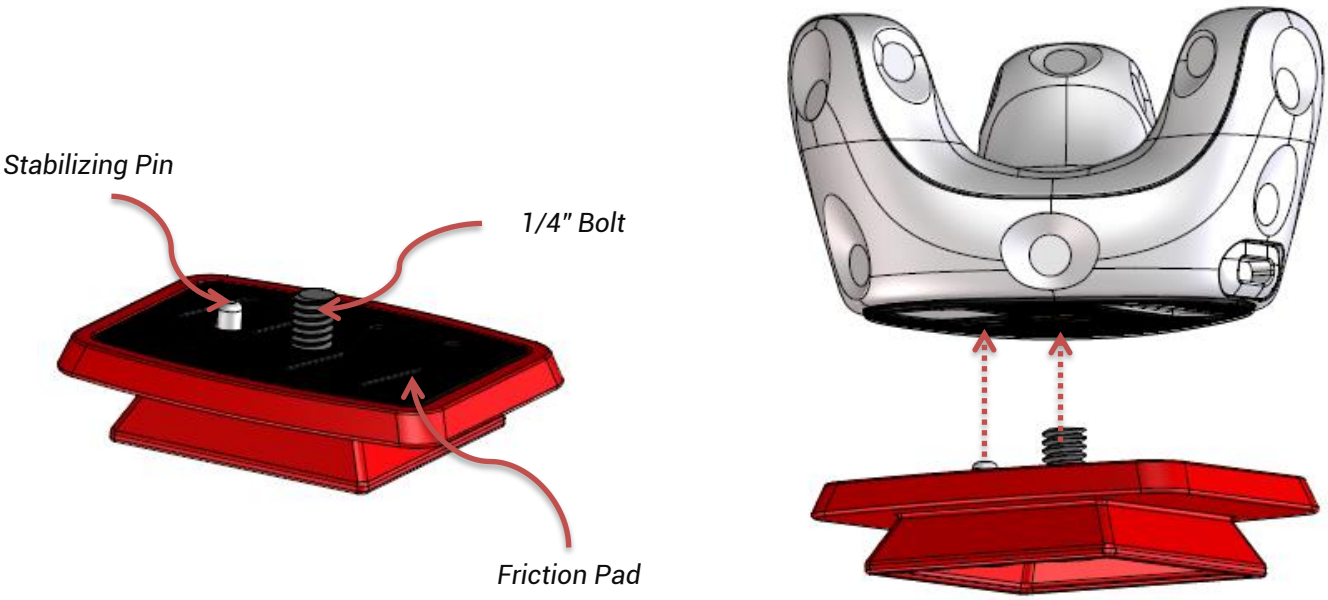
5. USB Port: Used for electrical connection to the accessory through a USB Type-C cable

Docking Mechanism

VIVE Tracker (3.0) applies the general camera tripod docking method, which follows ISO standards (ISO 1222:2010).

The following are the schematic drawings of how Tracker (3.0) can mount on an accessory.

Docking with standard tripod cradle head



VIVE Tracker (3.0) can be mounted on the cradle head first, and then attached to the main body of the accessory (similar to how a camera is mounted on a tripod).

Damping mechanism

During research and actual usage, it has been observed that continuous vibration in VIVE Tracker (3.0) will affect the IMU performance, causing noticeable IMU drift. To address this, it is suggested to use a damping system with the docking mechanism. The illustration below is a reference for how existing damping rubbers (purchased as a drone accessory) can serve this purpose.

When considering the vibration scales that the VR content plans to adopt, among the factors that can be adjusted are the durometer of the damper rubber, the mounting distance/position, and the usage of damper rubbers.

Accessory design

Below are the different accessory mechanisms that follow ISO standards:

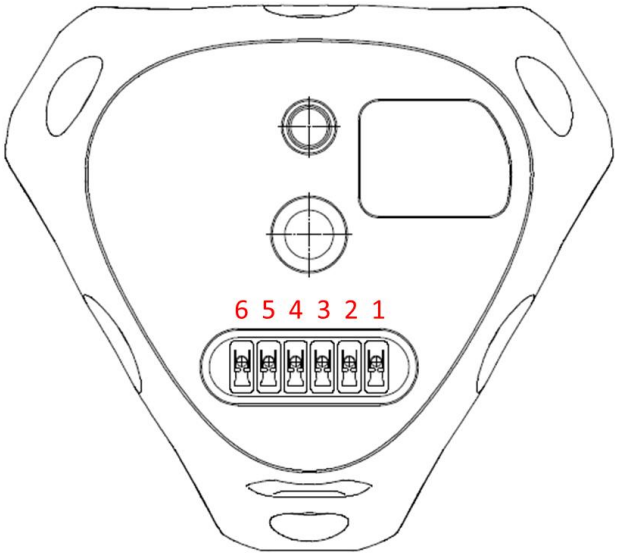
- 1/4" bolt design
  Please refer to ISO 1222-2010, Figure 1 on page 1.

- Stabilizing pin design
  VIVE Tracker (3.0) leverages the design from ISO 1222-2010, Figure 5 on page 3. For details on dimensions and tolerances, please refer to pages 13-17. It is suggested to apply the stabilizing pin for better tracking performance.

- Screw thread design
  The screw thread type that VIVE Tracker (3.0) uses 1/4" screw with 1.27 mm pitch. For detailed information, please refer to ISO 1222-2010, pages 3-5.

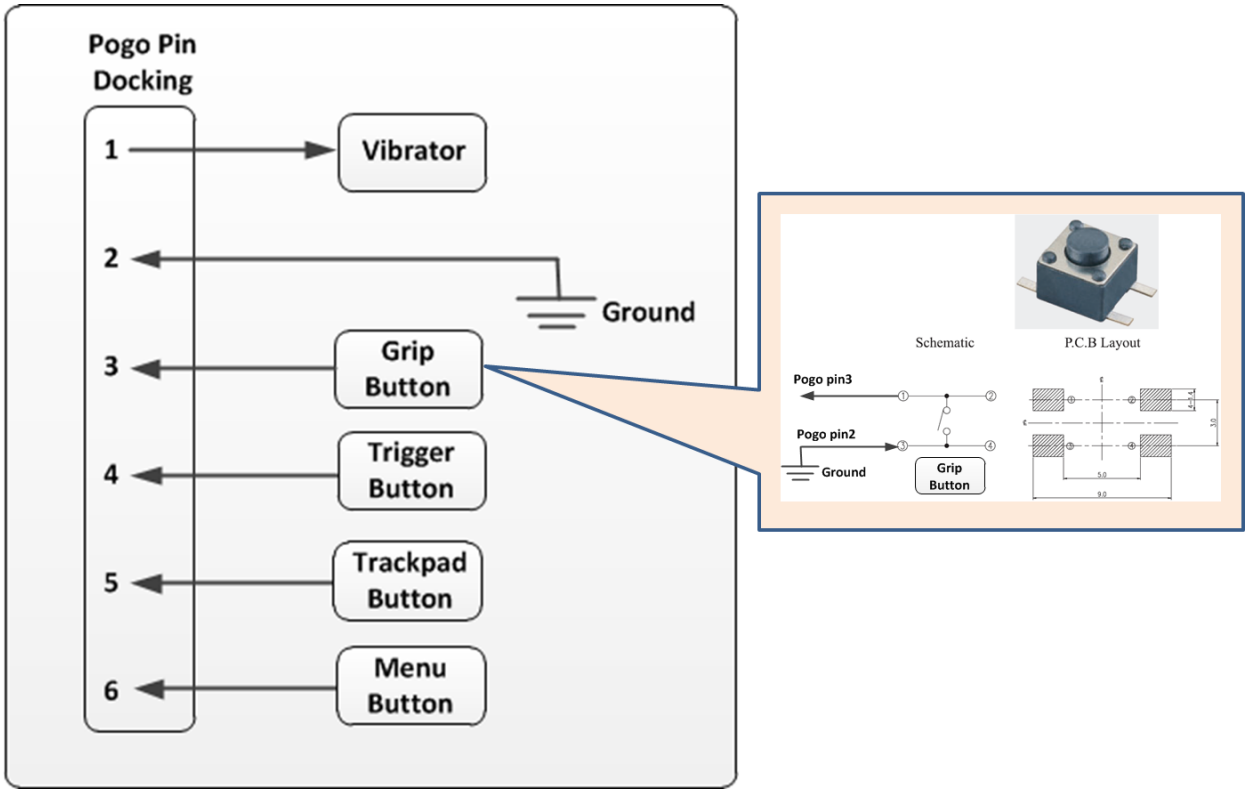
Pogo Pin design

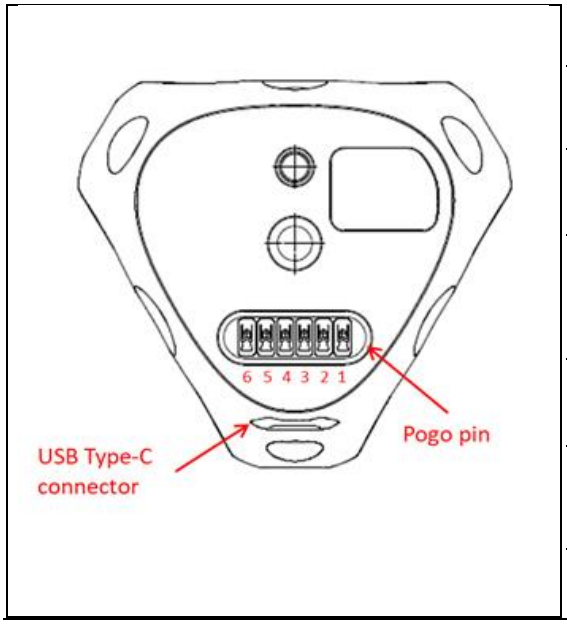
a. Pin definition of VIVE Tracker (3.0)



b. Pogo Pin reference design

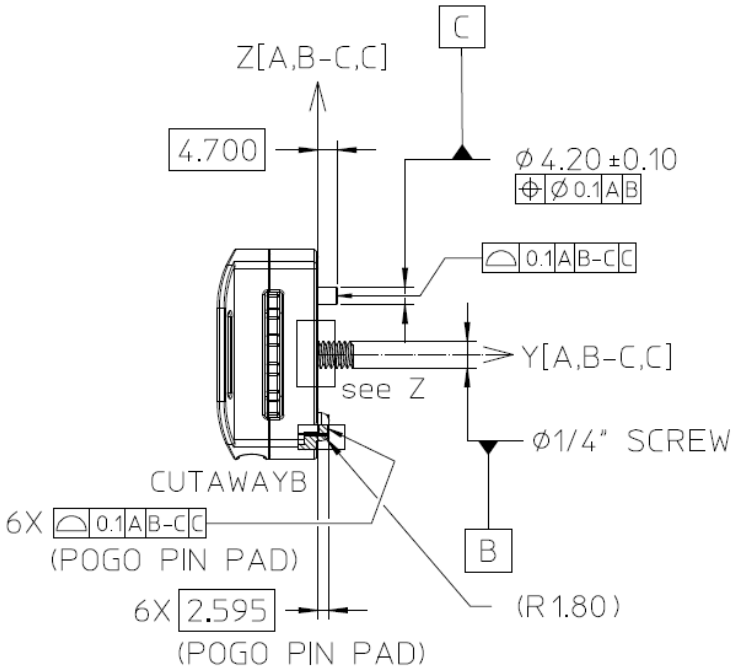
Electrical

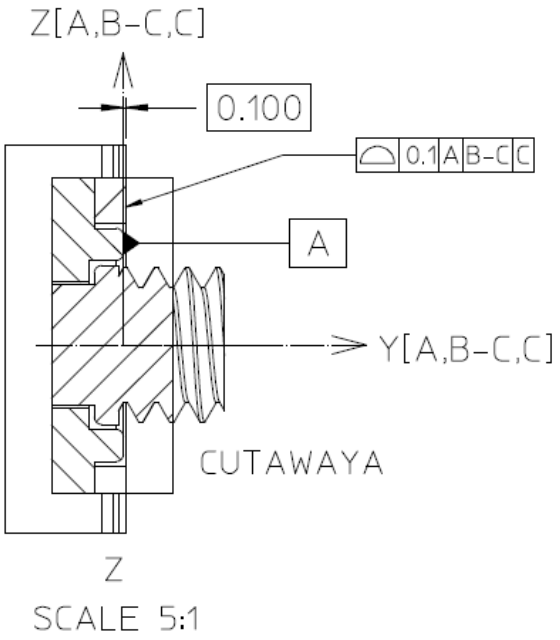


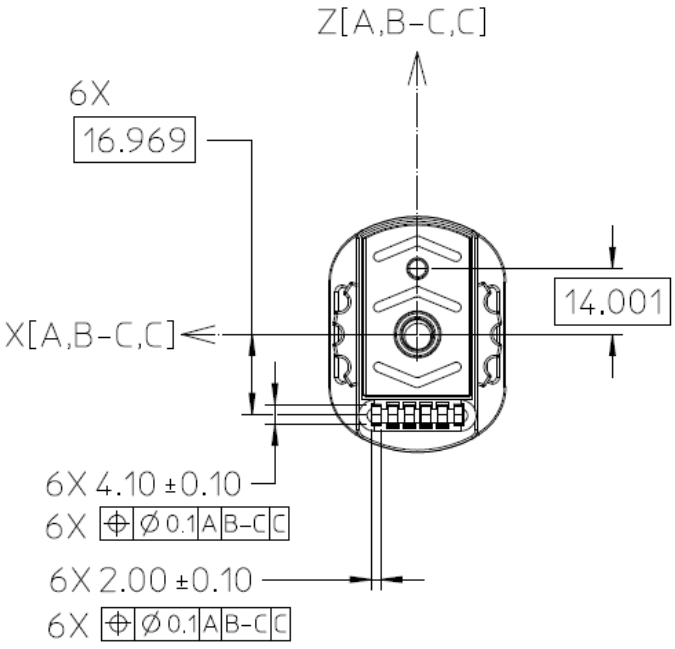


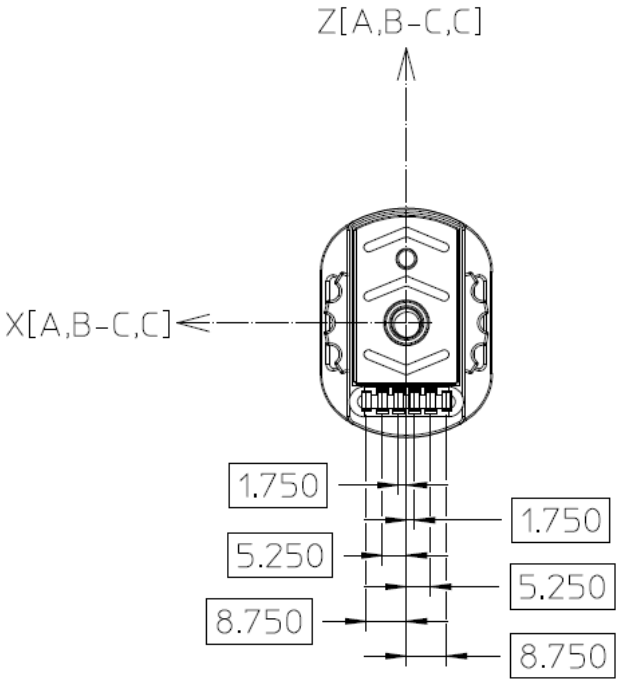
Pin no.	Type	Description
1	Digital output	General purpose output pin
2	GND	Ground
3	Digital/Power input	1. General purpose input pin: Internal pull up resistor to VDD, Active -low (Grip button) 2. Power input pin
4	Digital input	General purpose input pin: Internal pull up resistor to VDD, Active -low (Trigger button)
5	Digital input	General purpose input pin: Internal pull up resistor to VDD, Active -low (Trackpad button)
6	Digital input	General purpose input pin: Internal pull up resistor to VDD, Active -low (Menu button)

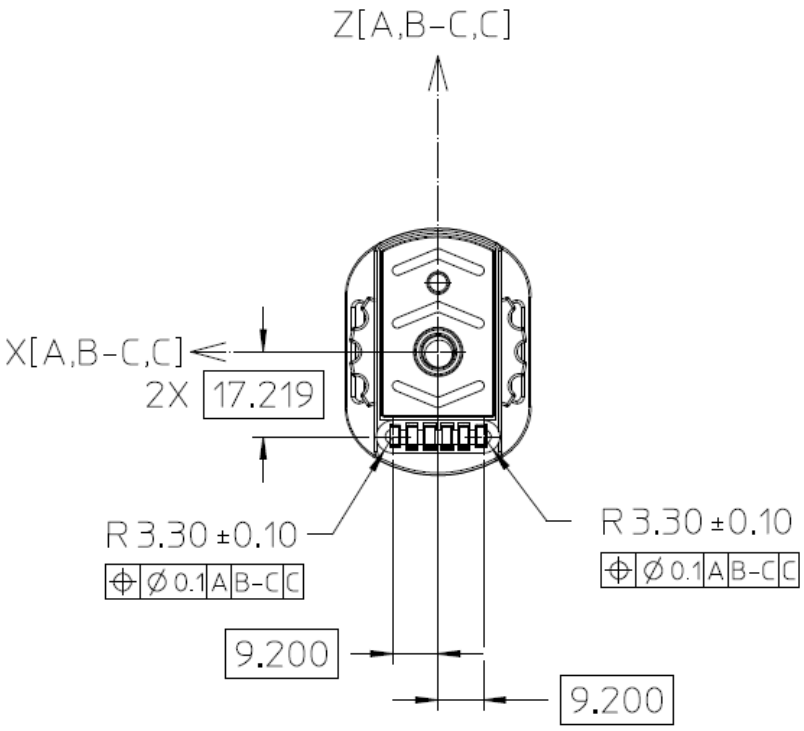
Mechanical

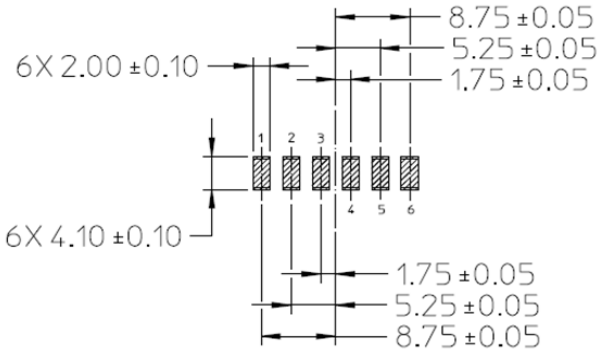

C
Z[A,B-C,C]
4.700
ø 4.20 ±0.10
ø 0.1 A B
0.1 A B-C C
Y[A,B-C,C]
see Z
ø1/4" SCREW
CUTAWAYB
6X 0.1 A B-C C
(POGO PIN PAD)
B
6X 2.595
(R 1.80)
(POGO PIN PAD)


Z[A,B-C,C]
0.100
0.1 A B-C C
A
Y[A,B-C,C]
CUTAWAYA
Z
SCALE 5:1


Z[A,B-C,C]
6X
16.969
14.001
X[A,B-C,C]
6X 4.10 ±0.10
6X Ø 0.1 A B-C C
6X 2.00 ±0.10
6X Ø 0.1 A B-C C


Z[A,B-C,C]
X[A,B-C,C]
1.750
1.750
5.250
5.250
8.750
8.750





Contact Resistance :

30mΩ (Max) initial when measured at 20mV (Max) open circuit at 100mA.

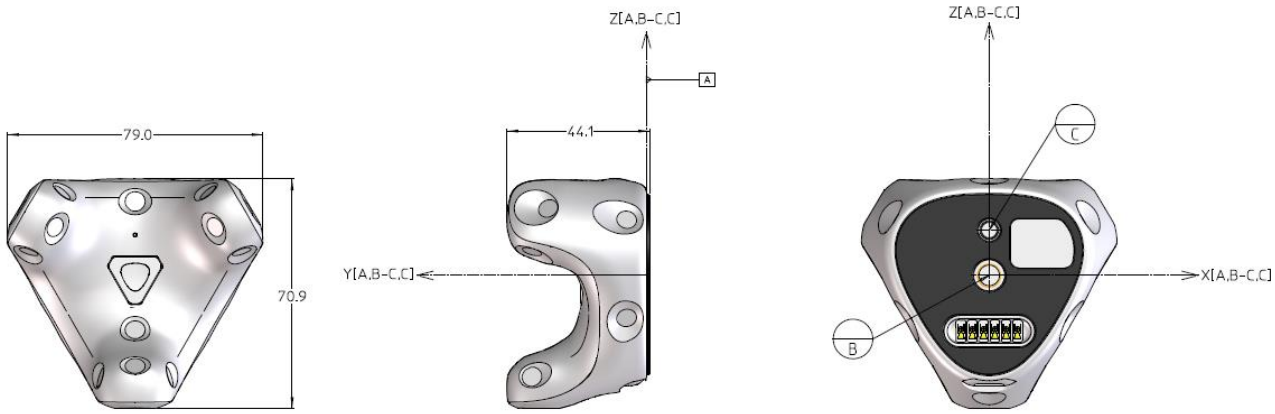
Contact Current Rating :

1.8A (Min)

Coordinate system

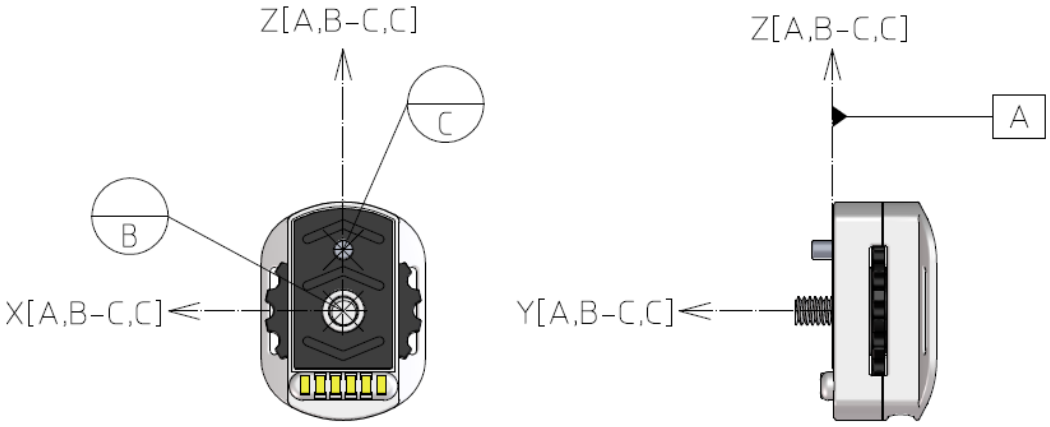
VIVE Tracker (3.0) uses the "**Right-handed coordinate system**".

- Datum A is set to be the top surface of the ring feature around the 1/4" Screw Nut.
- Datum B is set to be the intersection point between the centerline of Standard Camera Mount (1/4" Screw Bolt) and Datum A.
- Datum C is set to be the intersection point between the centerline of Stabilizing Pin Recess and Datum A.
- The coordinate system is constructed by the Datum frame of Datum A, the line of Datum B and Datum C, and Datum C itself.



Accessory:

- Datum A is set to be the top surface of the ring feature around the 1/4" Bolt.
- Datum B is set to be the intersection point between the centerline of 1/4" Screw and Datum A.
- Datum C is set to be the intersection point between the centerline of Stabilizing Pin and Datum A.
- The coordinate system is constructed by the Datum frame of Datum A, the line of Datum B and Datum C, and Datum C itself.



Software components

This section describes software components for the VIVE Tracker (3.0).

If you are an accessory maker, you can transfer a button event through the VIVE Tracker (3.0) Pogo pin port. You can refer to detailed data format transfer between an accessory and VIVE Tracker (3.0) in the Data Formats section.

If you are a content developer, refer to Unity integration.
Accessory integration.

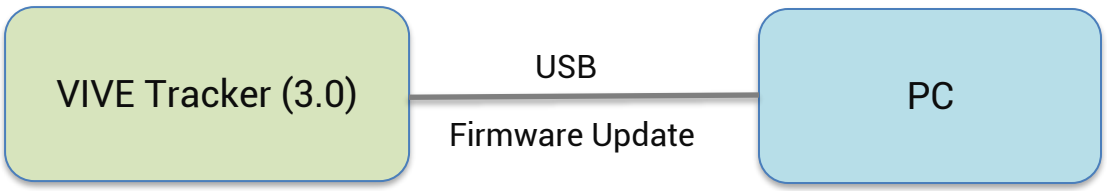
When new firmware for VIVE Tracker (3.0) is released, you can update the firmware by connecting the tracker to the PC using a USB Type-C cable. Learn how to update the firmware in Firmware update.

System requirements

For both content developers and accessory makers:

1. To test VIVE Tracker (3.0) with your content or accessory, you need to have HTC VIVE as well as the required hardware and software to run it.

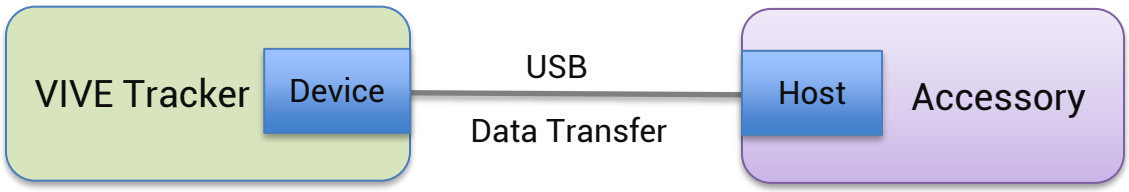
2. You need to have a PC with at least one available USB 2.0 port to plug in the dongle (for use cases with the dongle mentioned in Use Cases) or VIVE Tracker (3.0) (to update firmware). This PC should also SteamVR installed.



For accessory makers:

If your accessory needs to simulate buttons of the VIVE controller to a PC through VIVE Tracker (3.0), it must support the following interfaces:

- **Pogo pin:** Refer to hardware requirement section.
- **USB interface:** USB full speed host and HID class. The VIVE tracker (3.0) will act as a USB device to transfer data to/from the attached accessory.



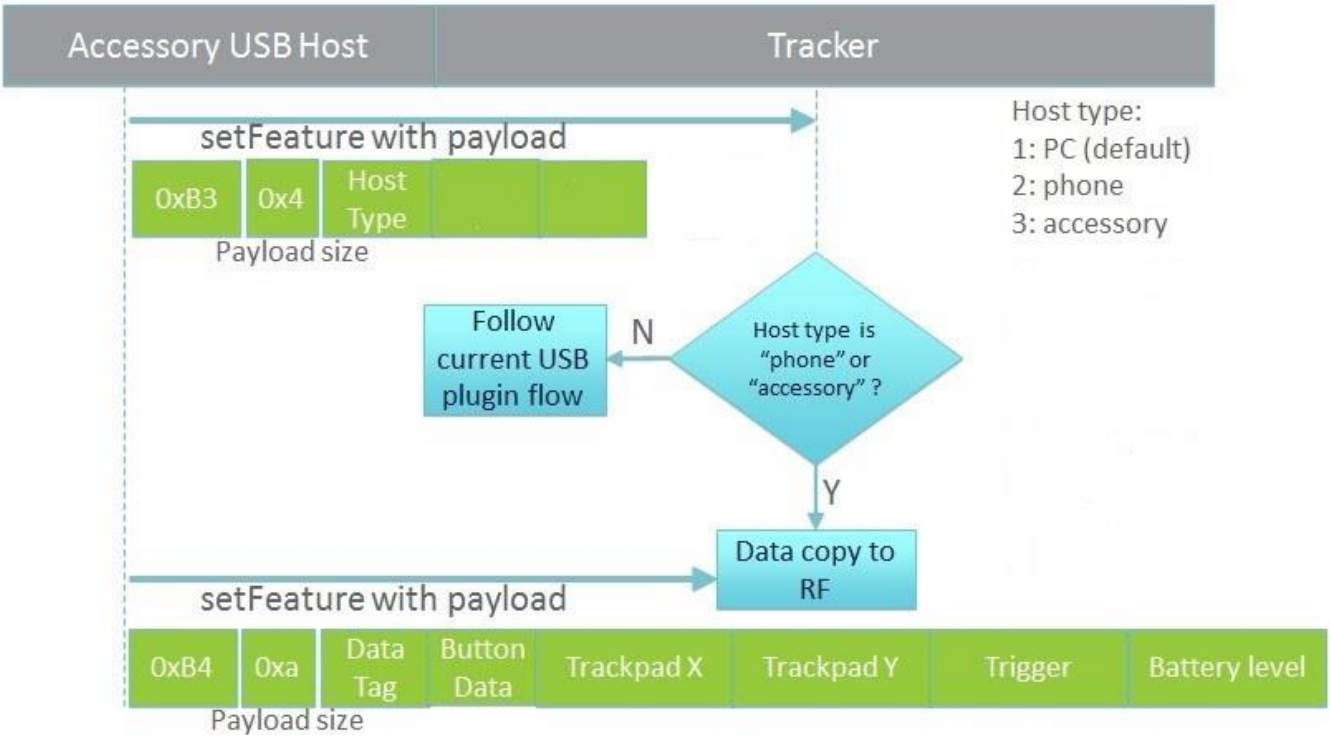
Data formats

This section describes the data formats that accessory makers can use to transfer data between the accessory and the PC through VIVE Tracker (3.0) when the USB interface is used.

The data format for transfer from an accessory to VIVE Tracker (3.0) is sent by a USB HID feature report. It is similar to the user interface of the VIVE controller. The interval to send data should be longer than 10 ms.

To learn more about the USB HID spec, please refer to information on the official USB website VIVE Tracker (3.0) will have three USB nodes when connected as a USB device to a PC. Use USB VID/PID as **28DE/2300** for VIVE Tracker (3.0), and check that the interface name changes to 'Controller'. Set the windex to **2** to send USB commands to VIVE Tracker (3.0).

Refer to the table below for the USB command flow between accessory and VIVE Tracker (3.0).



SetFeature 0xB3 data format

Byte Index	Data	Remark
0	Host Type	2 : phone 3 : accessory
1	N/A	Reserved
2	N/A	Reserved

SetFeature 0xB4 data format

Byte Index	Data	Remark
0	Tag Index	Indicates the version of the data being sent out. Default value is zero in this version of data format.
1	Button	TRIGGER 0x01 BUMPER 0x02 MENU 0x04 STEAM 0x08 PAD 0x10 PAD_FINGERDOWN 0x20 Reserved 0x40 Reserved 0x80

Byte Index	Data	Remark
2	Pad X value	Pad X value, value from -32768 to 32767 BYTE 2 is LSB
3		
4	Pad Y value	Pad Y value, value from -32768 to 32767 BYTE 4 is LSB
5		
6	Trigger Raw	Trigger Raw, value from 0 to 65535 BYTE 6 is LSB
7		
8	Battery Level	Battery Level, Reserved BYTE 8 is LSB
9		

Table: Data Format, Accessory to VIVE Tracker (3.0)

Sample code

Below are sample code to send the setFeature command to VIVE Tracker (3.0). However, you need to reference your system to have the correct API to send the USB setFeature command.

Visual Studio

```
buffer[0] = 0xB3;
buffer[1] = 0x03; // Length
buffer[2] = 0x03; // 1: PC, 2: Phone, 3: Accessory
buffer[3] = 0x01;
buffer[4] = 0x00;

if (!HidD_SetFeature(m_hDevice, buffer, sizeof(buffer))) {
      AfxMessageBox(L"Error: Failed to set feature.");
}
```

JAVA

```
// Take 0xB3 command for example
data1[0] = (byte) 179; //0xB3
data1[1] = 3;  //Means there are 3 bytes follow
data1[2] = 3; //Host Type: ACCESSORY
data1[3] = 1; //Reserved
data1[4] = 1; //Reserved

int result = mDeviceConn.controlTransfer(0x21, 0x09, 0x0300, 2, data1,
data1.length, 0);
```

STM F4 series developer board

```
char buffer[64];
char *p_buffer = buffer;
unsigned int buffer_size = sizeof(buffer);
memset( (void *)p_buffer, 0, buffer_size );
p_buffer[0] = 0xB3; // Command
p_buffer[1] = 0x03; // Size of Data
p_buffer[2] = 0x03; // Host Type: ACCESSORY
p_buffer[3] = 0x01;
p_buffer[4] = 0x01;

// STM API
USBH_HID_SetReport(
            handle,
            0x03,     // Feature Report
            0,
            (uint8_t *)p_buffer,
            buffer_size );
```

Accessory integration

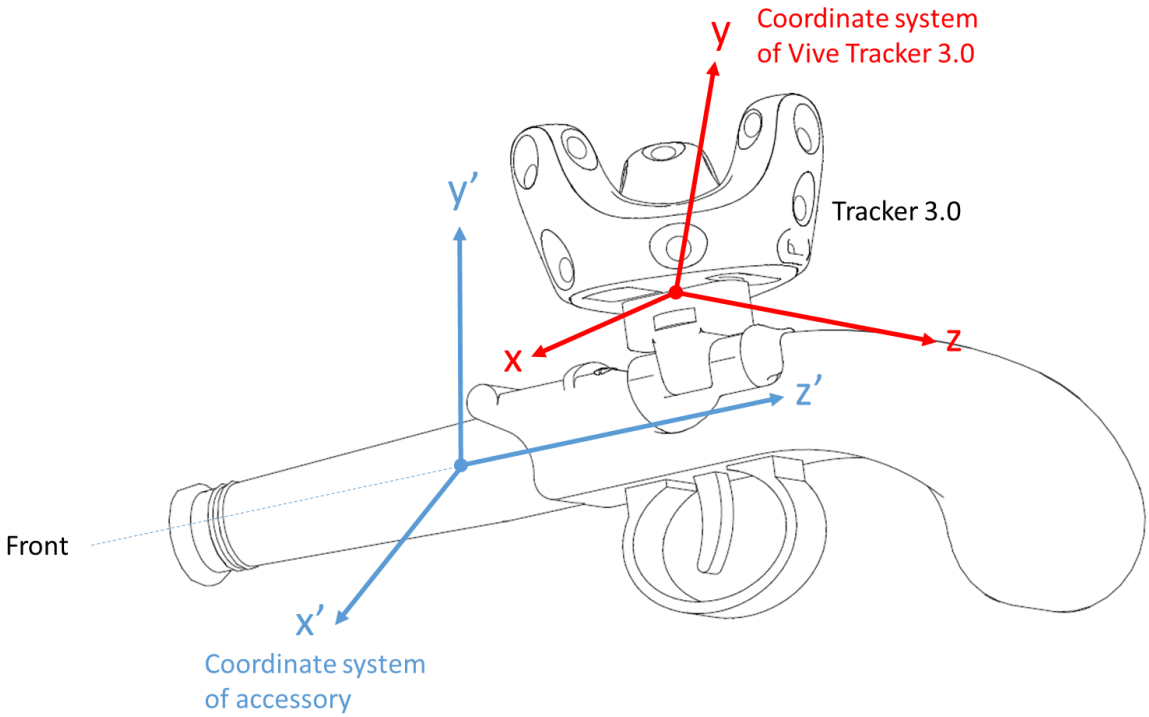
For Pogo out pin signal duration:
VIVE Tracker (3.0) receives haptic input value from the content, while the Pogo out pin will output HIGH with the duration value in “ms”.

This section describes information on position transformation between an accessory and VIVE Tracker (3.0). Content developers can create the correct rotation and translation result of the content used with the attached accessory in a game engine such as Unity.

It is assumed that the local coordinate system of the accessory is that z-axis is facing the front (left-handed coordinate system), and VIVE Tracker (3.0) is attached in the accessory as in the example below. Rotation degree and translation distance of an accessory relevant to VIVE Tracker (3.0) are described in roll, yaw, pitch and D_x, D_y, D_z respectively during the integration.

After the center of an accessory has been decided during the design, the following degrees and distance of an accessory based on actual integration condition can be measured. For detailed information regarding the center of the VIVE Tracker (3.0), refer to guidelines related to the hardware and mechanical design.

An example of integrating a gun accessory is described with VIVE Tracker (3.0):



Pitch: Angle that rotates around x-axis.
Yaw: Angle that rotates around y-axis.
Roll: Angle that rotates around z-axis

D_x: Center distance of x-axis between accessory and tracker.
D_y: Center distance of y-axis between accessory and tracker.
D_z: Center distance of z-axis between accessory and tracker.

Content developers can collect the above information and transform Tracker pose to accessory pose.

Assume Tracker rotation matrix is $R_{Tracker}$, accessory rotation matrix $R_{Accessory}$ = $R_{Pitch\_Yaw\_Roll} * R_{Tracker}$.

And accessory position $V_{Accessory}$= $V_{Tracker}$+ $R_{Accessory}$*$Distance$

The following is a Unity sample code (reference):

```
public class Accessory : MonoBehaviour {

    const float dX =  0.0100224f;
    const float dY = -0.07616526f;
    const float dZ =  0.4884118f;

    const float roll  = 10.854305f;
    const float yaw   = 91.8736f;
    const float pitch = 78.805113f;

  void Update () {

        //Collect delta rotation and displacement between Tracker and
Accessory
        Vector3 delta_displacement = new Vector3(dX, dY, dZ);
        Quaternion delta_rotation  = Quaternion.Euler(roll, yaw, pitch);

        //Get current Tracker pose
        Vector3 tracker_position    =
SteamVR_Controller.Input(3).transform.pos;
        Quaternion tracker_rotation =
SteamVR_Controller.Input(3).transform.rot;

        //Transform current Tracker pose to Accessory pose
        GameObject.Find("Accessory ").transform.rotation =
tracker_rotation * delta_rotation;
        GameObject.Find("Accessory ").transform.position =
tracker_position + (tracker_rotation * delta_rotation) *
delta_displacement;

  }

}
```

Another Unity sample code shows how to transform the accessory by comparing vectors parallel to y-axis and z-axis of the VIVE Tracker (3.0) (AxisY_Tracker, AxisZ_Tracker in the example below) and the accessory (AxisY_Accessory, AxisZ_Accessory in the example below).

```
public class Accessory: MonoBehaviour {

    const Vector3 AxisY_Tracker = new Vectors(AxisY_Tracker_X,
AxisY_Tracker_Y, AxisY_Tracker_Z);
    const Vector3 AxisZ_Tracker = new Vectors(AxisZ_Tracker_X,
AxisZ_Tracker_Y, AxisZ_Tracker_Z);

    const Vector3 AxisY_Accessory = new Vectors(AxisY_Accessory_X, AxisY_
Accessory _Y, AxisY_ Accessory _Z);
    const Vector3 AxisZ_ Accessory = new Vectors(AxisZ_Accessory_X,
AxisZ_ Accessory _Y, AxisZ_ Accessory _Z);

  void Update () {

        //Calculate delta rotation by comparing vectors parallel to Y
axes of Tracker and the accessory
        Quaternion delta_rotY = Quaternion.FromToRotation(AxisY_Tracker,
AxisY_Accessory);
        AxisZ_Tracker = delta_rotY * AxisZ_Tracker;
        Quaternion delta_rotZ = Quaternion.FromToRotation(AxisZ_Tracker,
AxisZ_Accessory);

        //Collect delta rotation and displacement between Tracker and
Accessory
        Vector3 delta_displacement = new Vector3(dX, dY, dZ);
        Quaternion delta_rotation  = delta_rotZ * delta_rotY;

        //Get current Tracker pose
        Vector3 tracker_position    =
SteamVR_Controller.Input(3).transform.pos;
        Quaternion tracker_rotation =
SteamVR_Controller.Input(3).transform.rot;

        //Transform current Tracker pose to Accessory pose
        GameObject.Find("Accessory").transform.rotation = delta_rotation
* tracker_rotation;
        GameObject.Find("Accessory").transform.position =
tracker_position + (delta_rotation * tracker_rotation) *
delta_displacement;
    }
`
```

Unity integration

This section provides an example for content developers to enable VIVE Tracker (3.0) in their VR content by using the Unity game engine.

First, VIVE Tracker (3.0) needs to be detected by SteamVR. Assuming that you have 2 VIVE controllers connected the dongle and dongle cradle is connected to the PC, right-click on a controller icon. In the menu, click **Pair Controller**. Then, press the Power button on VIVE Tracker (3.0) for 2 seconds to enter pairing mode.

After VIVE Tracker (3.0) successfully pairs with the dongle, you will see the VIVE Tracker icon in the SteamVR window.

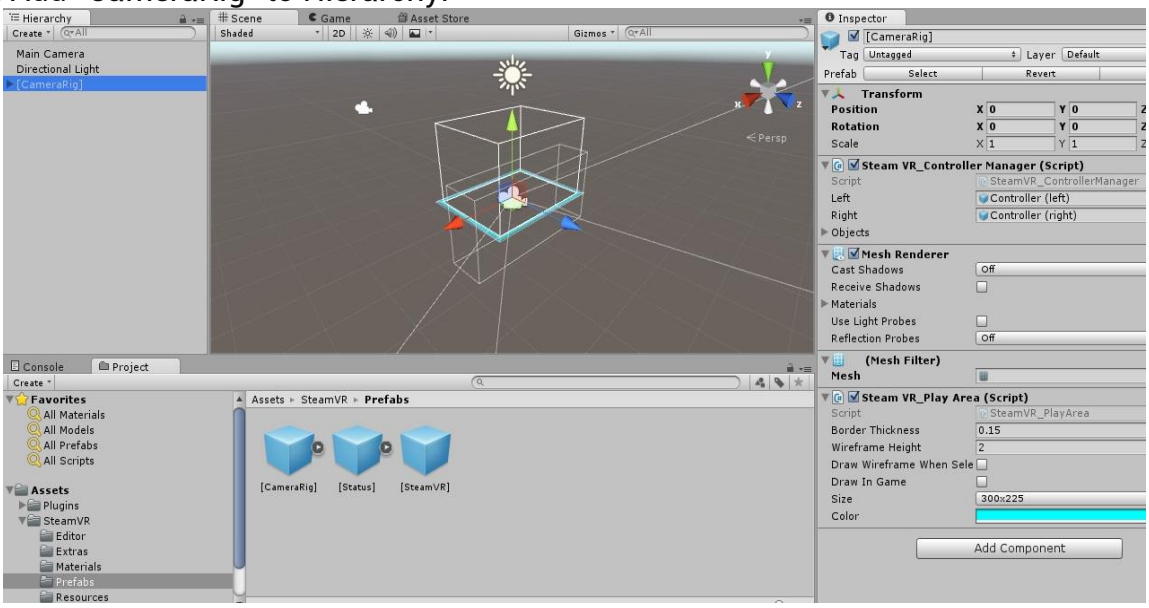
It is recommended that you use Unity version 2017.3 or newer (minimum version 5.4.6). You can download the latest one here.

Using the SteamVR plugin

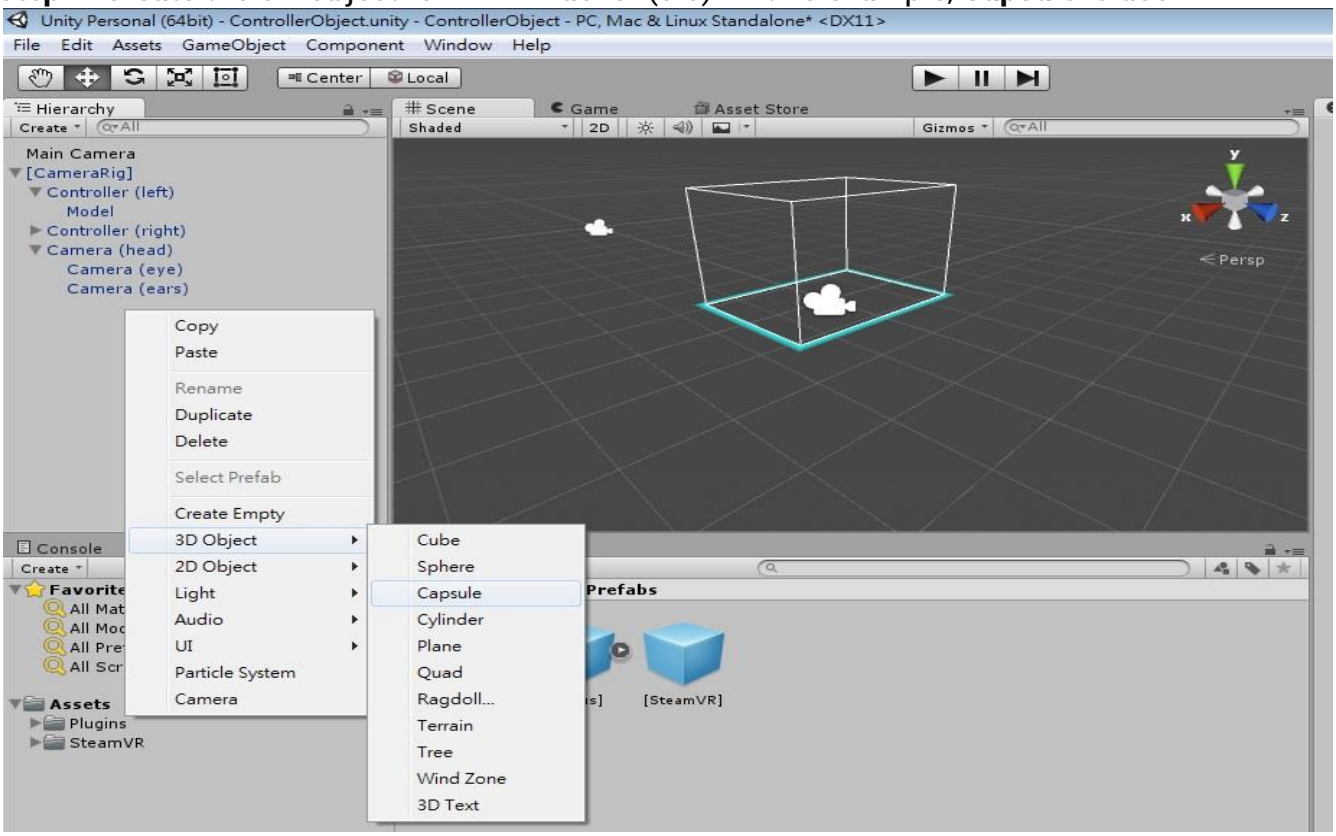
Import the SteamVR Plugin into your project. If you do not have the plugin, download it from the Unity Asset Store.

From the developer version of VIVE Tracker (3.0), it uses a similar approach and naming as when creating content for the VIVE controller. Follow these steps to create content for VIVE Tracker (3.0) in Unity.

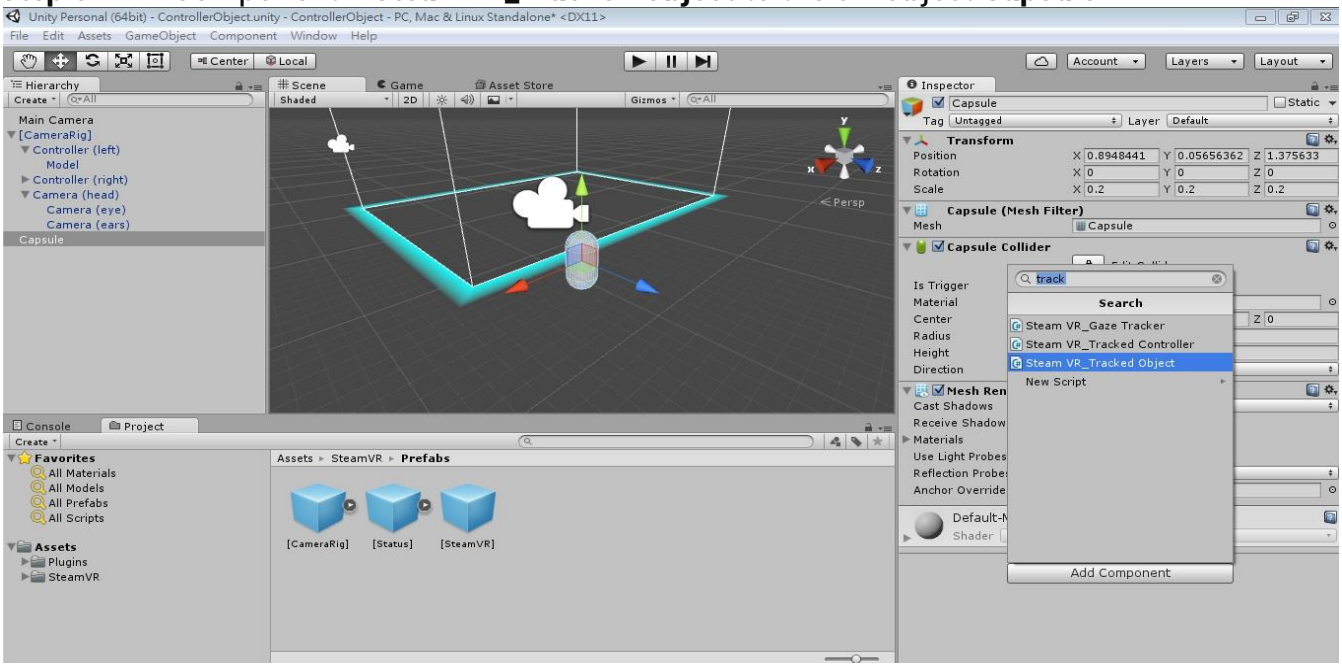
Step 1: Add “CameraRig” to Hierarchy.



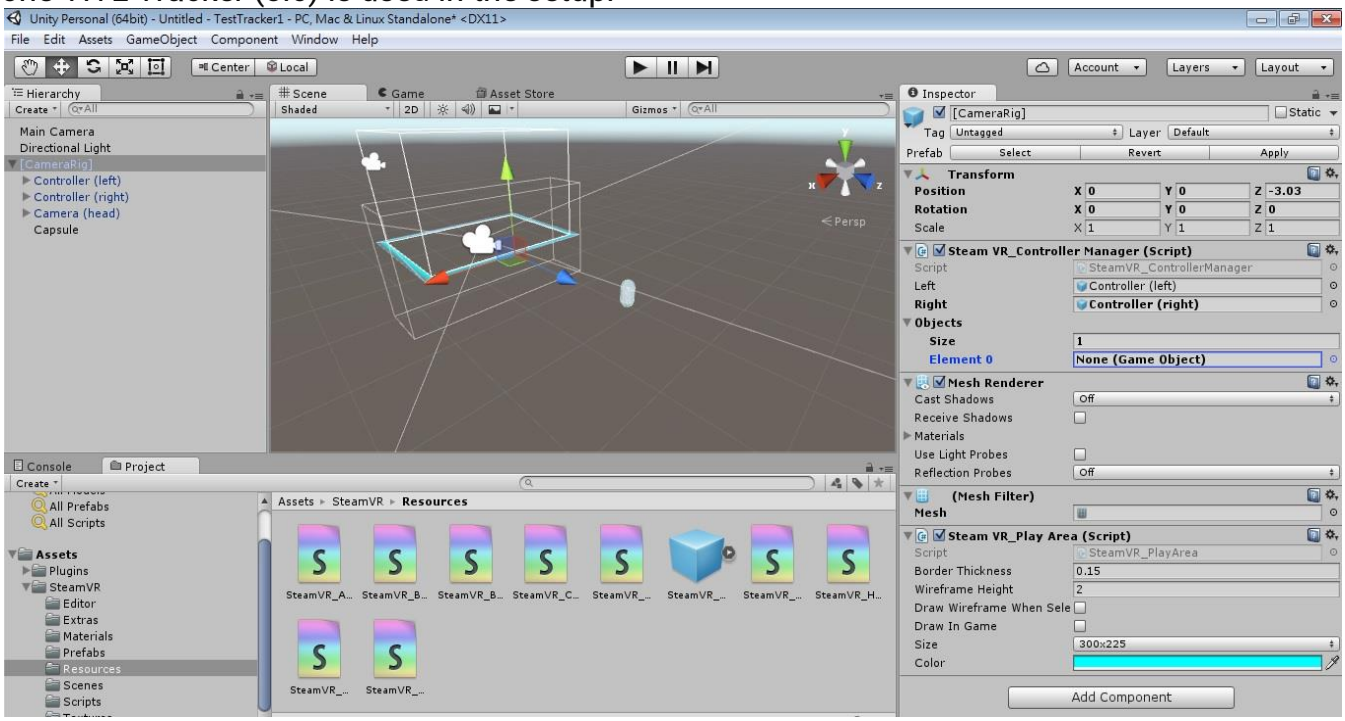
Step 2: Create the 3D Object for VIVE Tracker (3.0). In this example, **Capsule** is used.



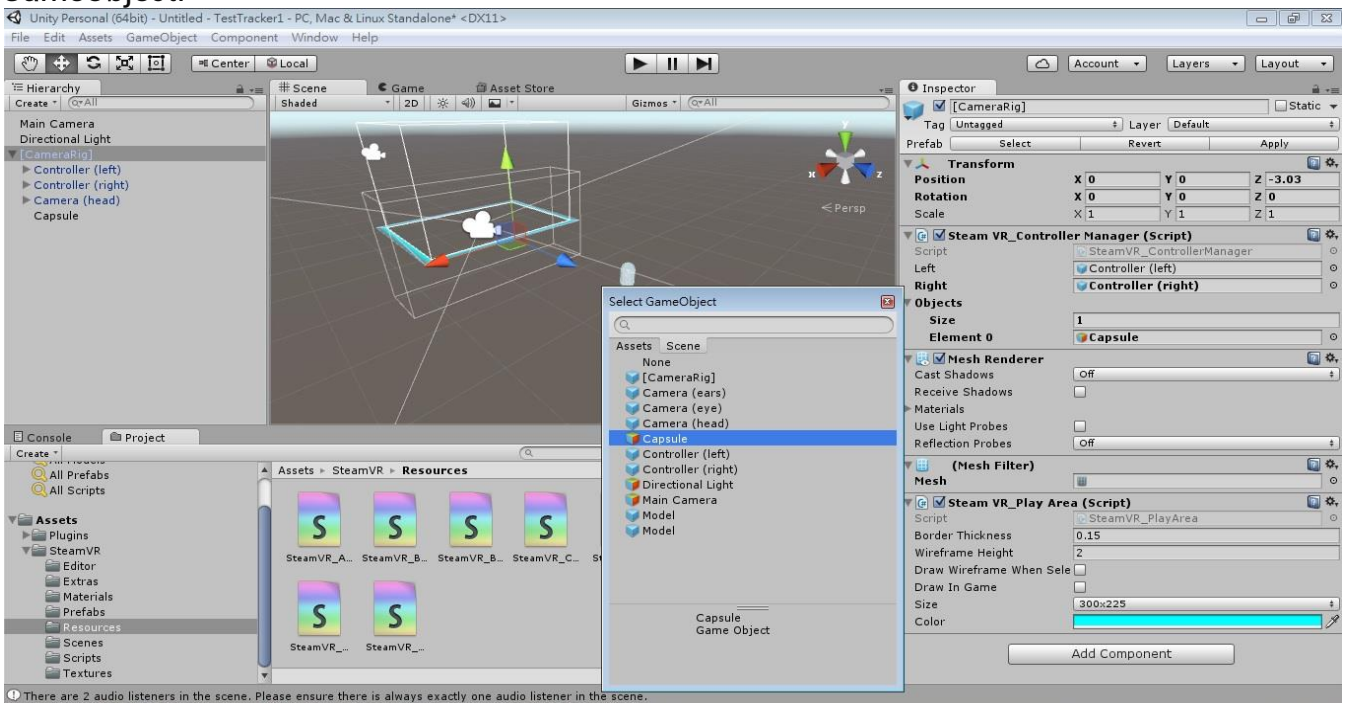
Step 3: Add Component > **SteamVR_Tracked Object** to the 3D Object **Capsule**.



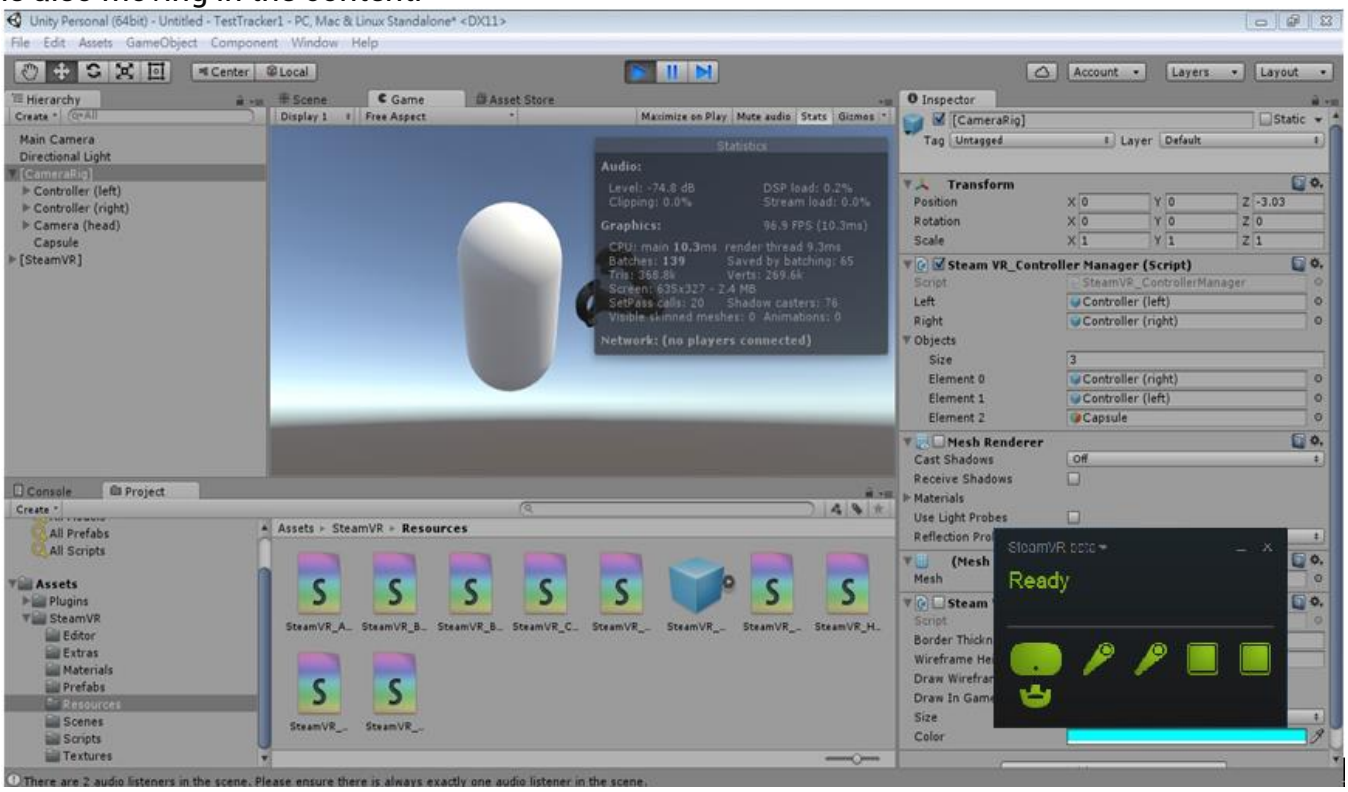
Step 4: Under **SteamVR Controller Manager**, set the size of the Objects item. In this example, one VIVE Tracker (3.0) is used in the setup.



Step 5: Under **SteamVR Controller Manager**, in the Element 0 field, select **Capsule** as the GameObject.

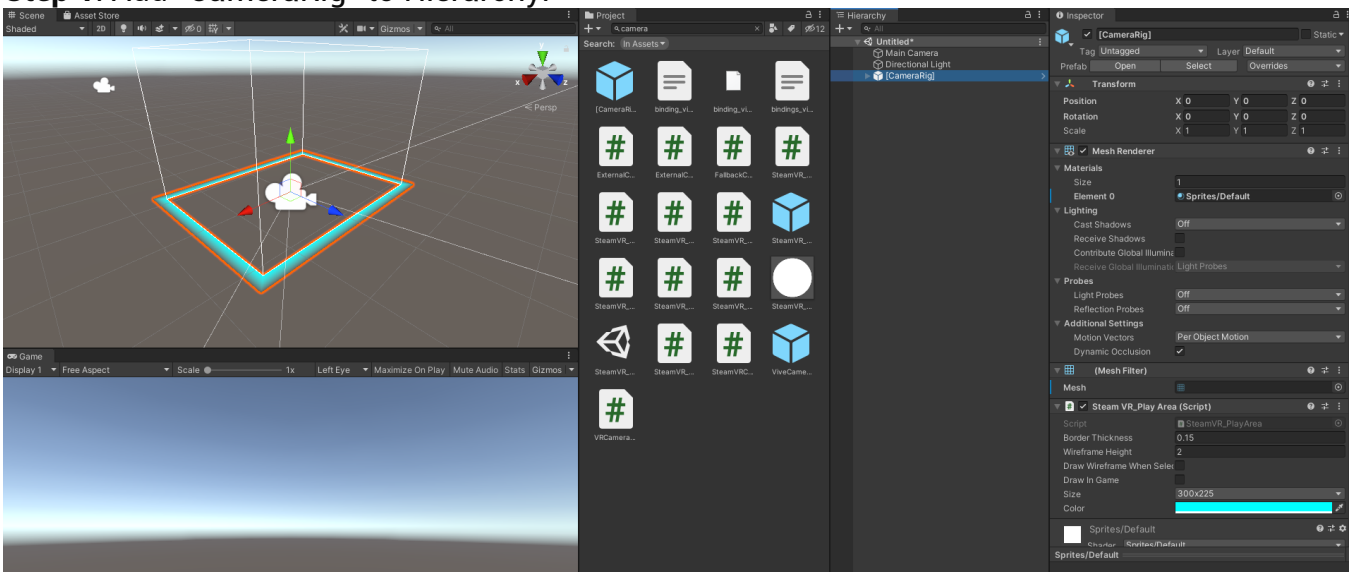


Step 6: In Unity, click **Run**. When you move VIVE Tracker (3.0), you will see the Capsule object is also moving in the content.

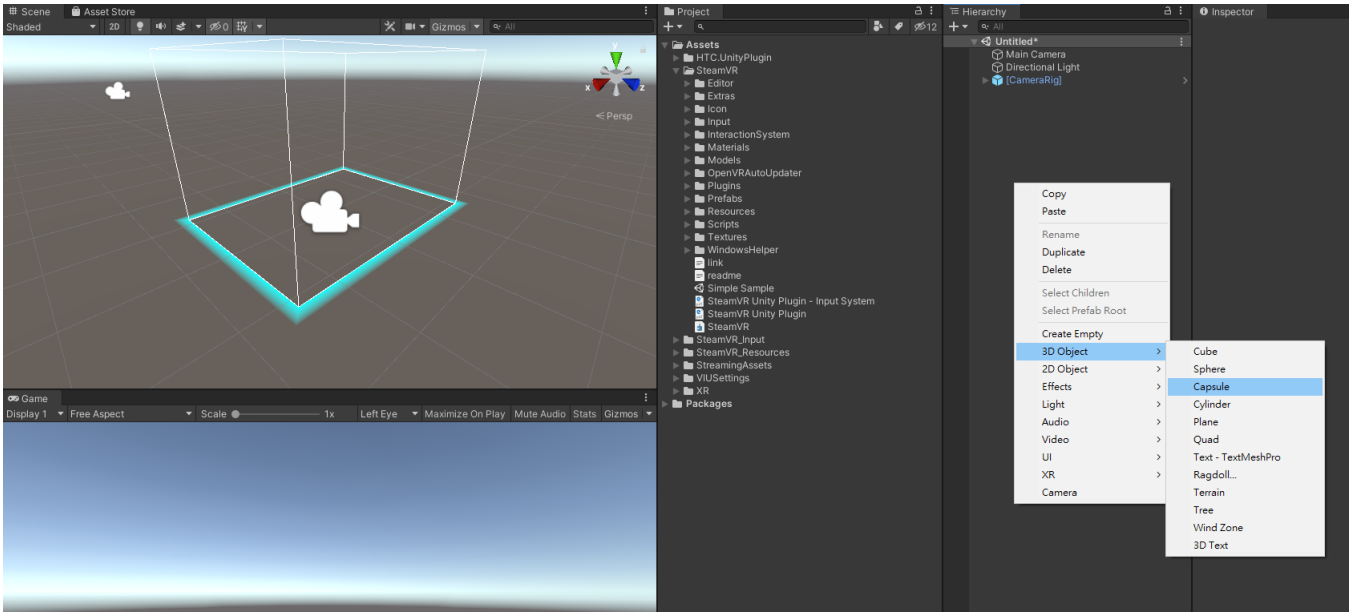


Using the SteamVR 2.0 plugin

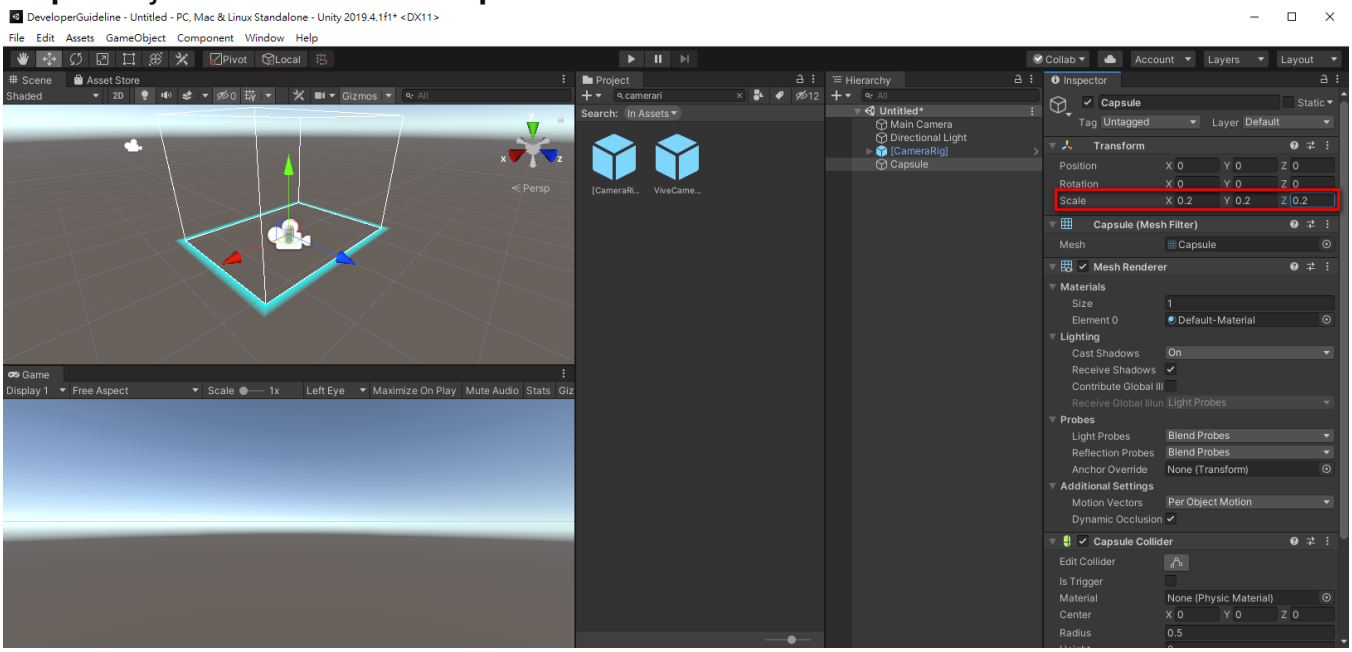
Step 1: Add "CameraRig" to Hierarchy.



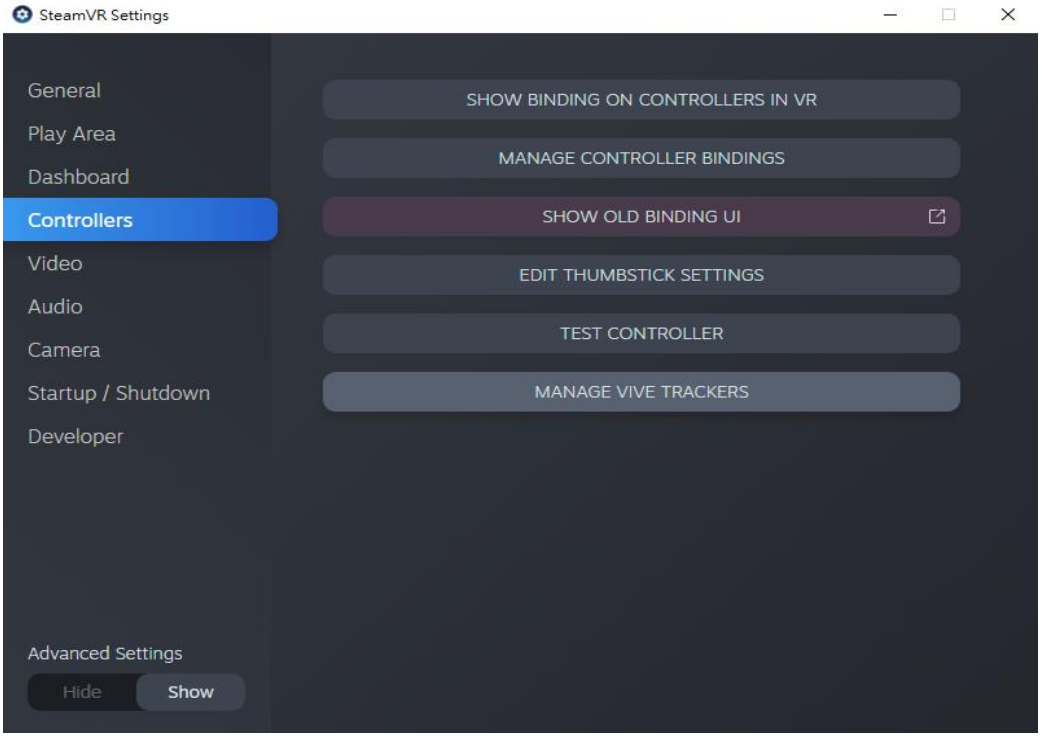
Step 2: Create the 3D Object for VIVE Tracker (3.0). In this example, **Capsule** is used.

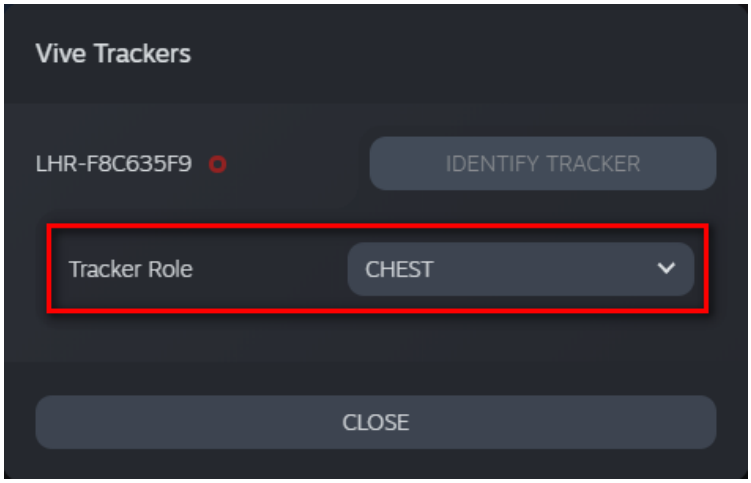


Step 3: Adjust the scale of the **Capsule** to 0.2.

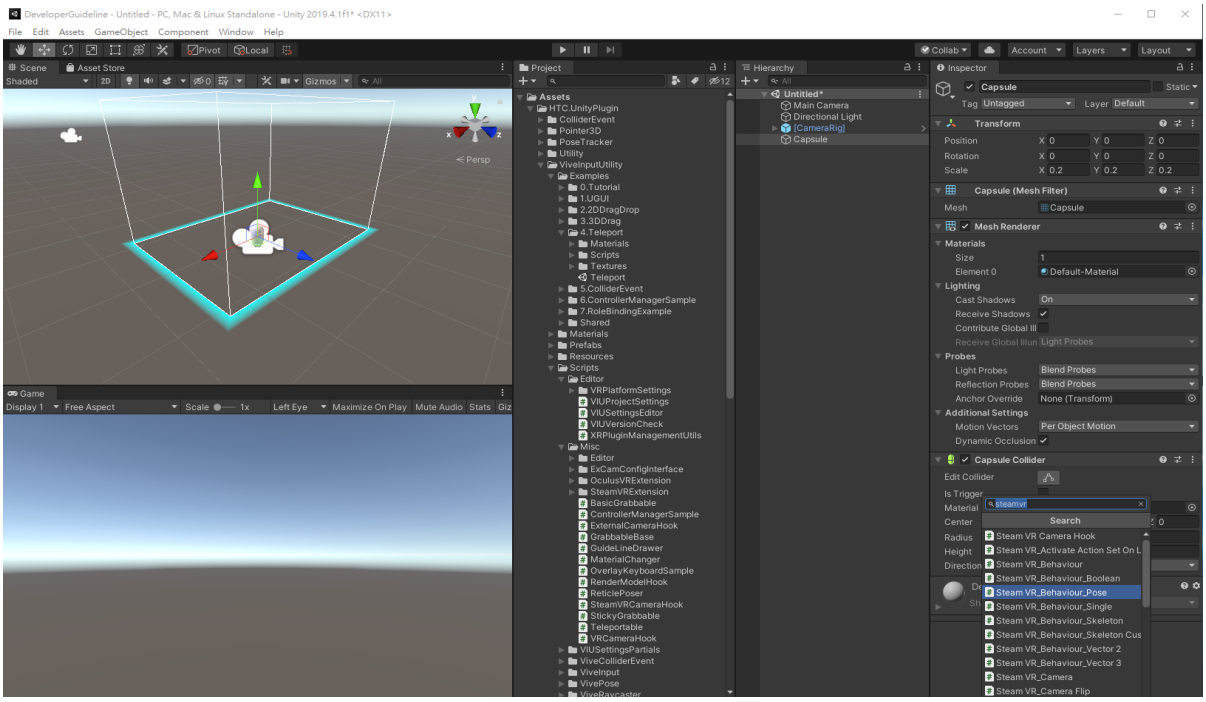


Step 4: In SteamVR, click **Devices** > **Controller Settings** > **Manage VIVE Trackers** to edit **Tracker Role** of your tracker. In this example, **Chest** is used.

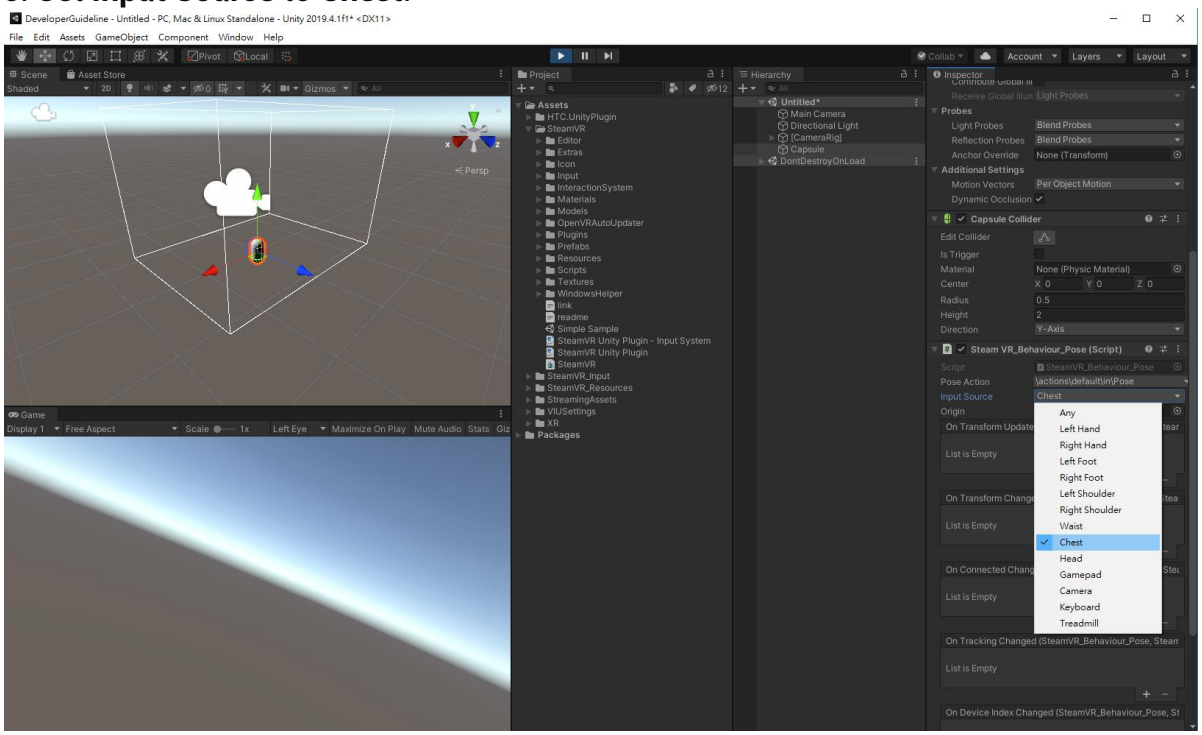




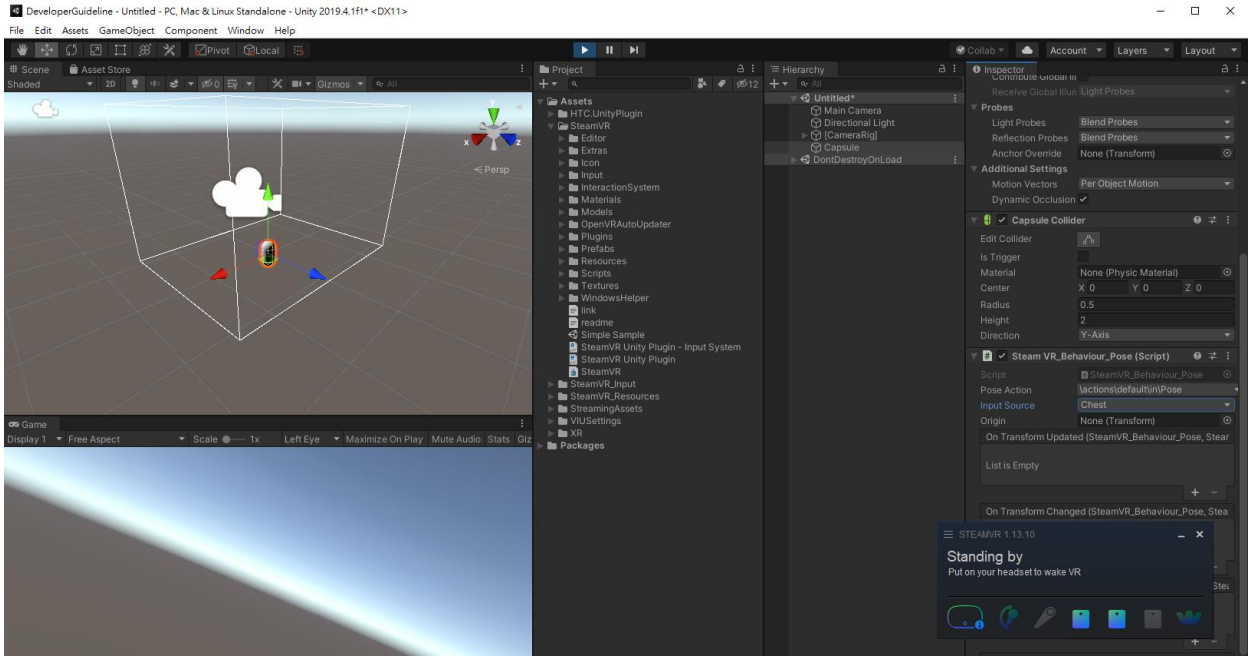
Step 5: Add **SteamVR_Behaviour_Pose o**bject



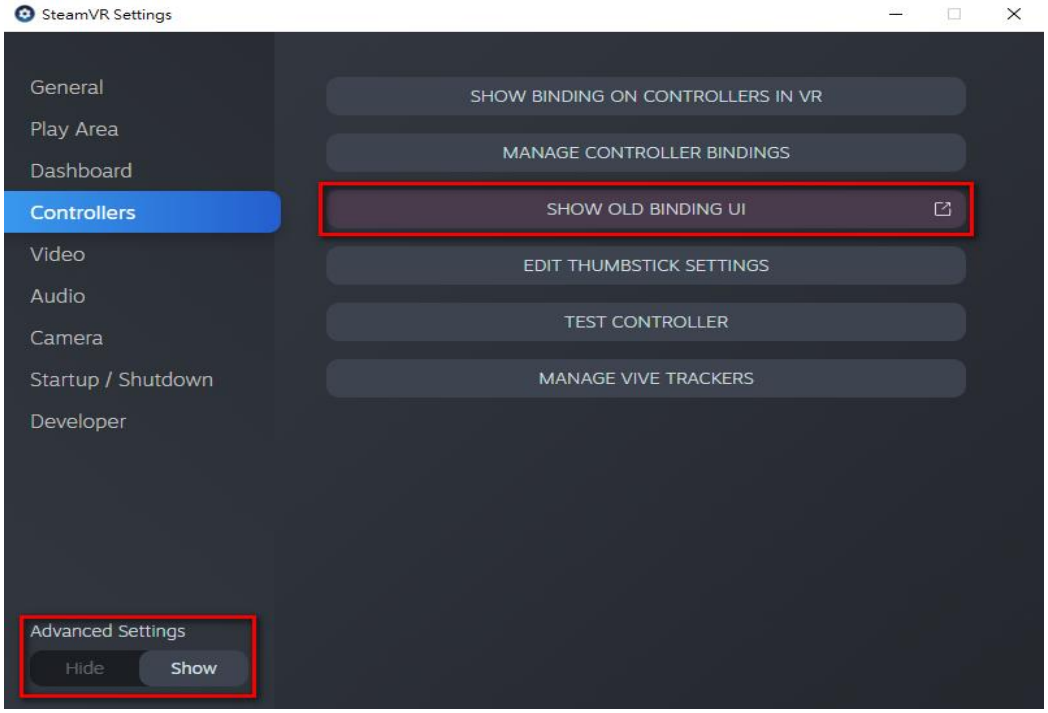
Step 6: Set **Input Source** to **Chest**.



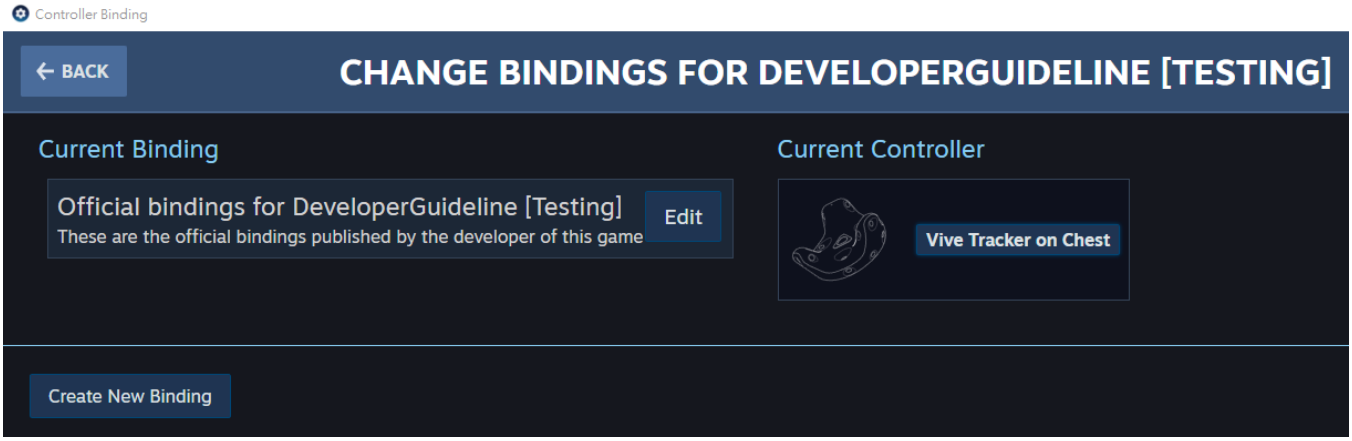
Step 7: In Unity, click **Run**. You will need to “Add Action Pose” for your tracker.



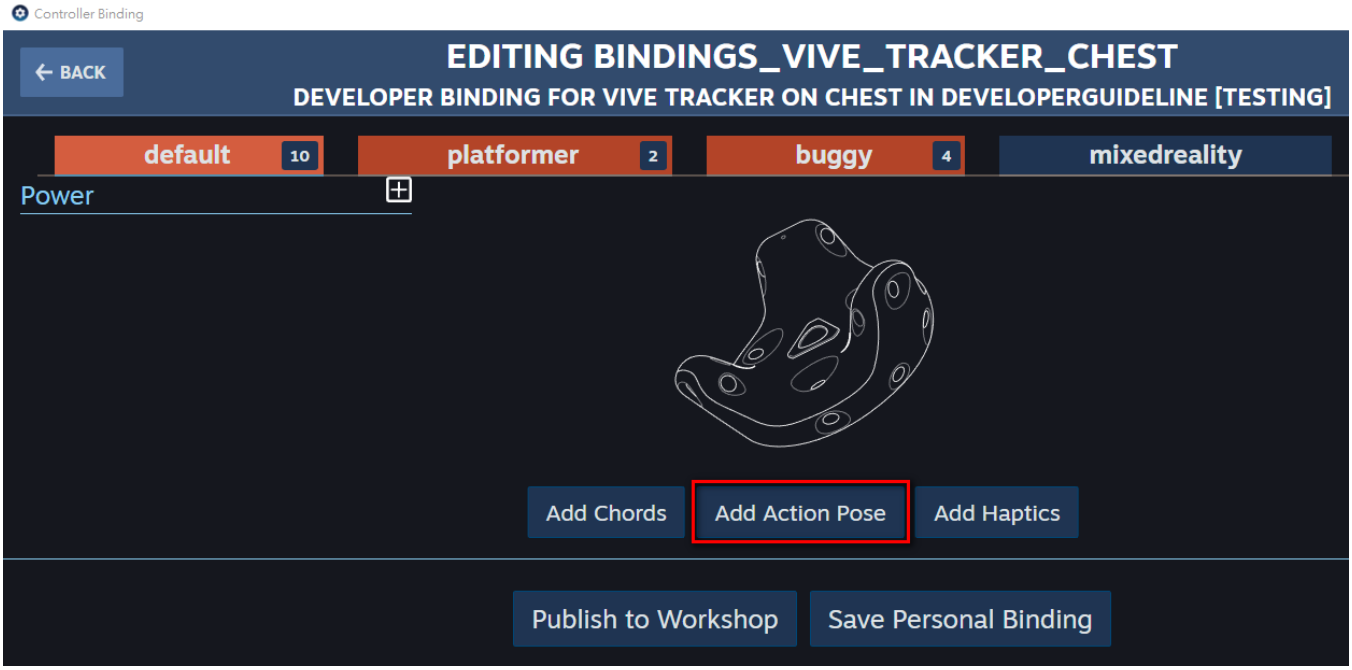
In SteamVR Settings, click **Show** in **Advanced Settings**. Click **Show Old Binding UI**, and then choose your project. In this example, **DeveloperGuideline** is used.



After selecting your application, click on **Edit** for **Vive Tracker on Chest**.

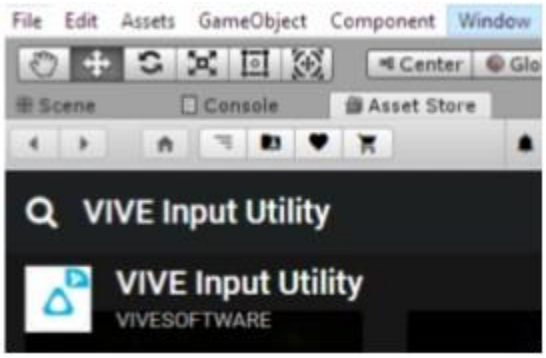


Click **Add Action Pose** and change the poses of */user/chest/pose/raw* from **Unused** to **Pose**. When you move VIVE Tracker (3.0), you will see the Capsule object is also moving in the content.

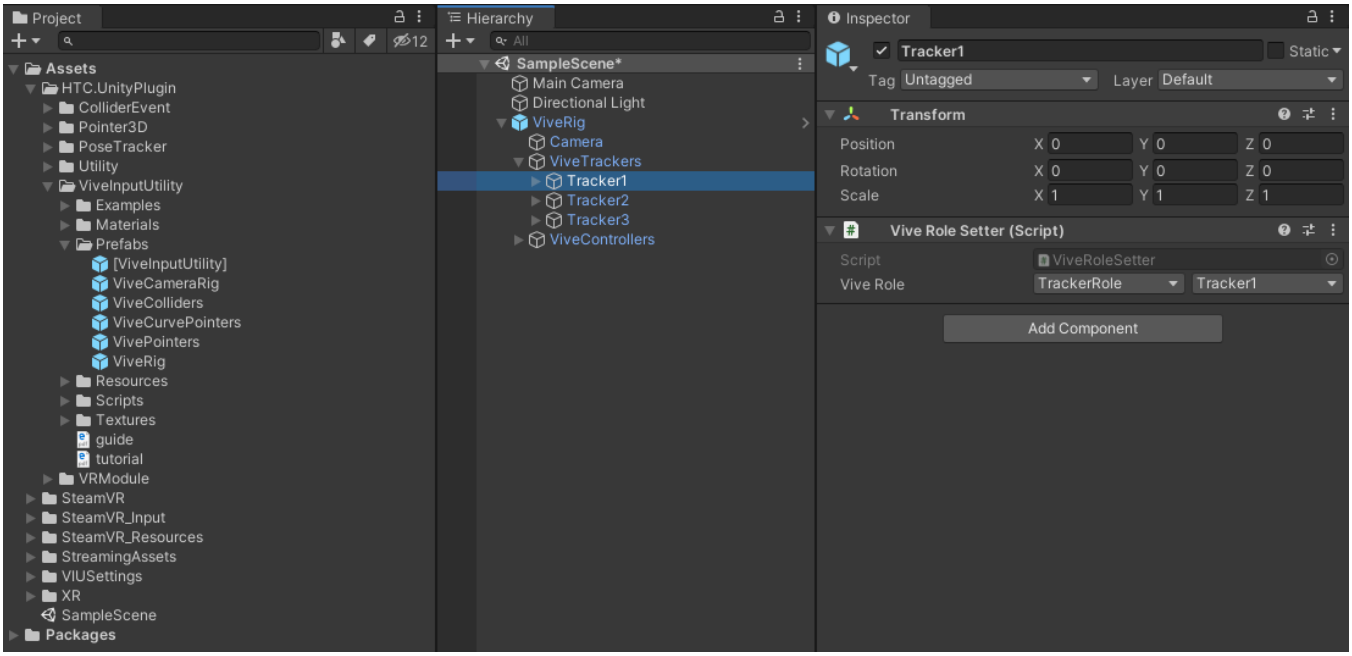


Using VIVE Unity plugin

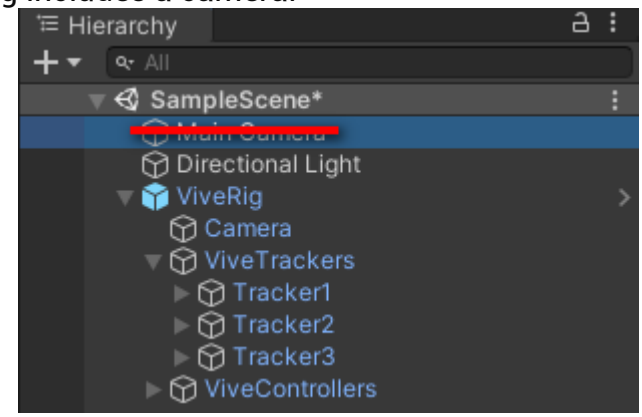
Import the VIVE Input Utility plugin into your project. If you do not have the plugin, download it from the Unity Asset Store. The VIVE Input Utility (VIU) plugin is a cross-platform VR toolkit with additional support for the VIVE Trackers.



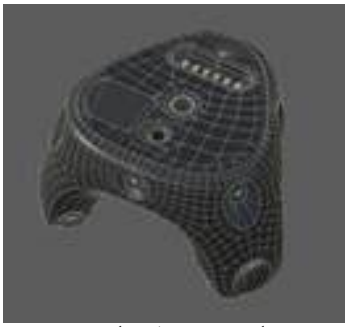
Step 1: Add “ViveCameraRig" or "ViveRig" to scene Hierarchy to add support for controllers and trackers in Unity.



Step 2. Just like SteamVR's "CameraRig", you can remove the existing "Camera" from the scene because the ViveRig includes a camera.



Step 3. Run to see the trackers supported just like the controllers. You will see the models from SteamVR if you have the SteamVR plugin, otherwise you will only see included models. The VIVERig and VIVECameraRig prefabs include support for up to three trackers. To add additional trackers, simply duplicate a tracker and rename the game object. In the inspector, make sure to also update the ViveRole.



SteamVR Tracker (3.0) model (example)

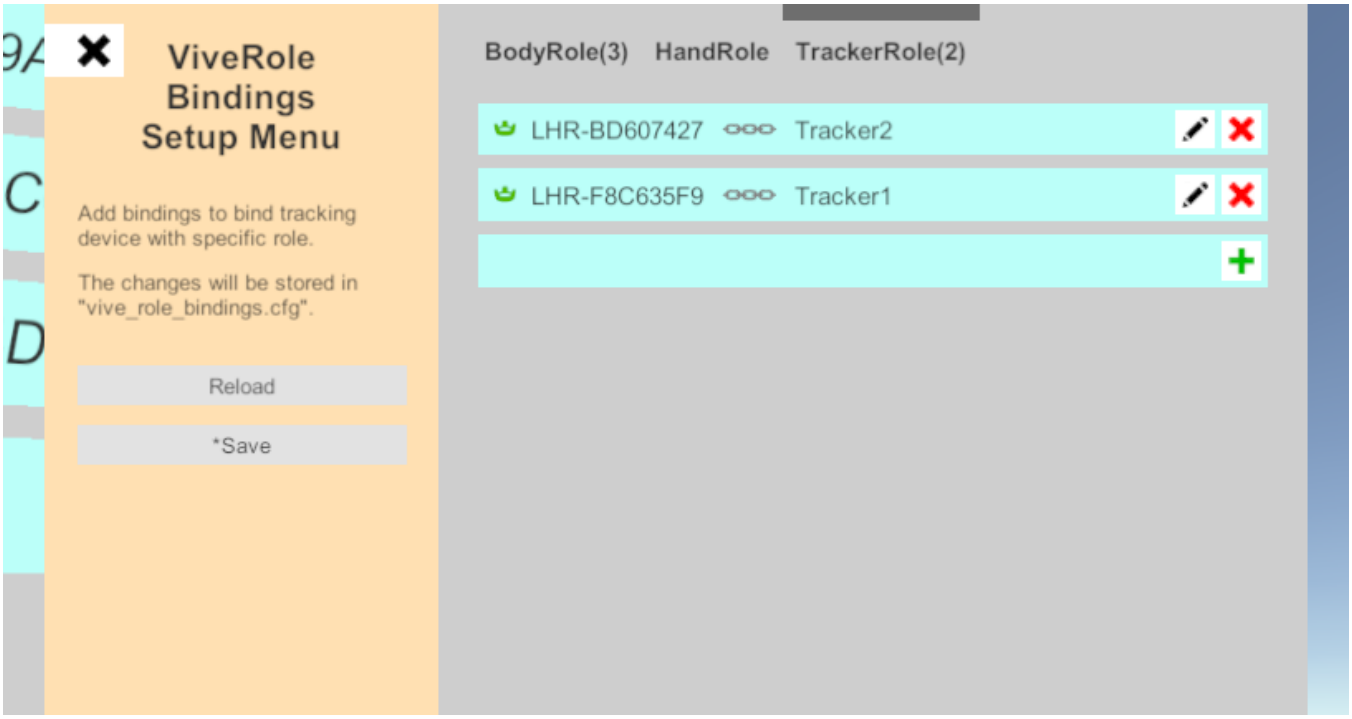
Note: Currently in both SteamVR and VIU, there is support for up to 11 (or 13, if not using controllers) trackers for a total of 16 devices (including the HMD and base stations). Recently, OpenVR has been updated to support up to 64 devices, but this has not been updated yet in the Unity plugins (but developers can add support with their own code).

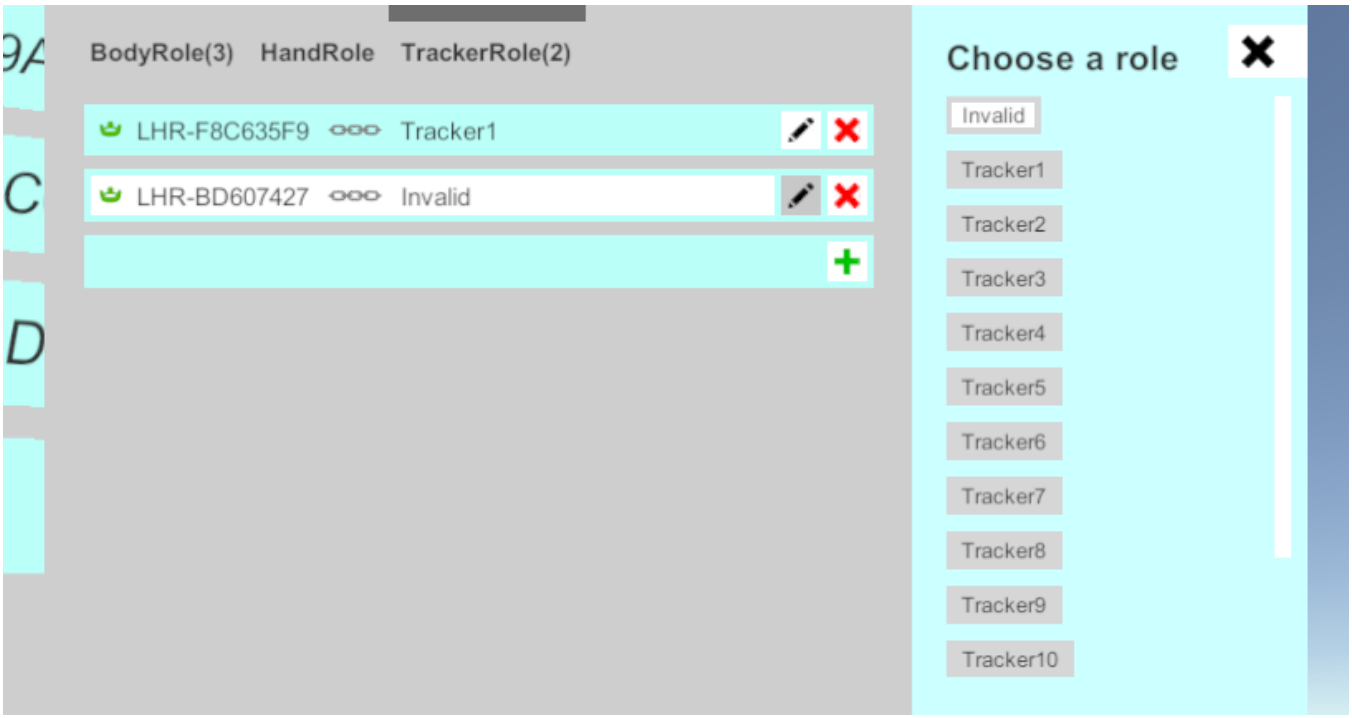
VIVE Trackers and VIVE Roles

One of the benefits of using the VIVE Input Utility is that you can swap VIVE trackers or redefine where they are attached to without worrying about managing device IDs and serial numbers.

You can now assign VIVE Roles depending on the context: TrackerRole, BodyRole, HandRole and DeviceRole. For example, you can define the same VIVE tracker as always assigned to your left foot using BodyRole LeftFoot. You can also assign for trackers to the same role to make it easier to swap out tracker devices when one is low on battery.

The VIVE Input Utility also provides a tool to discover and assign roles without additional code.
A keyboard shortcut, which can be assigned to display this overlay UI:





Additionally, there is an API available to create your own VR UI for assigning roles. For more information, check this github article. An example project is included in the VIU plugin - "7.RoleBindingExample.

Tracker on Unity or Unreal Engine

If you encounter problems in enabling VIVE Tracker (3.0) on Unity or Unreal Engine, refer to the following:

- For Unity 3D:
  - Download link for the VIVE Input Utility package: AssetStore or GitHub
  - VIVE Input Utility source code repository: https://github.com/VIVESoftware/VIVEInputUtility-Unity

- For Unreal Engine 4:
  - Check that you're using Unreal Engine version 4.26 or newer.

Firmware update

Check the SteamVR notification to update the firmware. You can update the VIVE Tracker (3.0) firmware by:

1. Copy the firmware binary files (including MCU, FPGA and RF) provided by HTC into the same folder of "lighthouse_watchman_update.exe" in your PC.
2. If a VIVE controller is connected to your computer through USB, unplug it first.
3. Connect one VIVE Tracker (3.0) to your computer using the USB cable.
4. Execute the following commands:

   a. Update the MCU firmware:
   ```
   lighthouse_watchman_update --target=default watchman_v3.fw
   ```

   b. Update the FPGA firmware:
   ```
   lighthouse_watchman_update.exe --target=default ice40_hdk_xxx.fw
   ```

   c. Update the RF firmware:
   ```
   lighthouse_watchman_update --target=default nrf52_xxx.fw
   ```