

# HTC Vive Tracker Developer Guidelines Ver. 1.4

# Version Control

Version Number	Version Date	Version Reason
1.0	2016.09.26	Initial version
1.1	2016.12.05	<ol style="list-style-type: none"><li>1. Use case image revised.</li><li>2. Pogo pin rearranged.</li><li>3. USB cable connection method deleted.</li><li>4. Data format revised.</li></ol>
1.2	2017.01.09	<ol style="list-style-type: none"><li>1. SteamVR monitor images revised.</li></ol>
1.3	2017.01.19	<ol style="list-style-type: none"><li>1. Design of Pogo Pin Pad revised.</li><li>2. Firmware upgrade revised.</li></ol>
1.4	2017.05.08	<ol style="list-style-type: none"><li>1. Docking revised.</li><li>2. Data format revised.</li><li>3. Tracker on Unity or Unreal added.</li><li>4. FAQ added.</li></ol>

# Contents

Introduction .....	1
Use cases .....	1
Hardware requirements .....	3
Interface .....	4
Radio frequency (RF).....	5
Power.....	6
Optics .....	6
Docking .....	7
Docking embodiments.....	7
Mechanical considerations.....	11
Apparel size.....	12
Main feature.....	13
Docking mechanism.....	14
Docking with standard tripod cradle head (w/o electric connection) .....	14
Docking with side tightening wheel (w/ electric connection if needed).....	14
Accessory design .....	15
Design of Pogo Pin Pad.....	16
Coordinate system .....	17
Software components.....	24
System requirements .....	24
Data format .....	26
Accessory integration .....	29
Unity integration .....	33
Tracker on Unity or Unreal .....	40
Firmware upgrade .....	41
FAQ.....	42

## Introduction

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

Vive Tracker 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 can:

- Simulate buttons of the Vive controller through the underlying Pogo pin.
- Send specific data to a PC via the USB interface of Vive Tracker or use its original approach to do it.

## Use cases

There are five use cases supported by Vive Tracker.

**Use Case 1:** Track passive objects with a USB cable in VR. In this case, the dongle is not used, and Vive Tracker connects with a PC by USB directly to transfer tracking data.

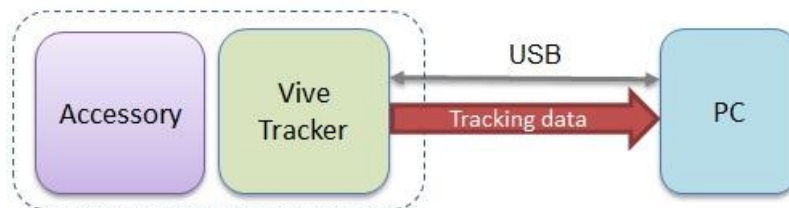


Figure: Use case 1 of Vive Tracker

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

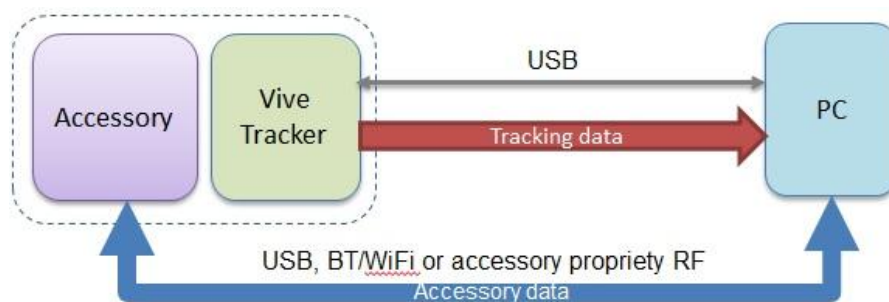


Figure: Use case 2 of Vive Tracker

**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 to a PC.

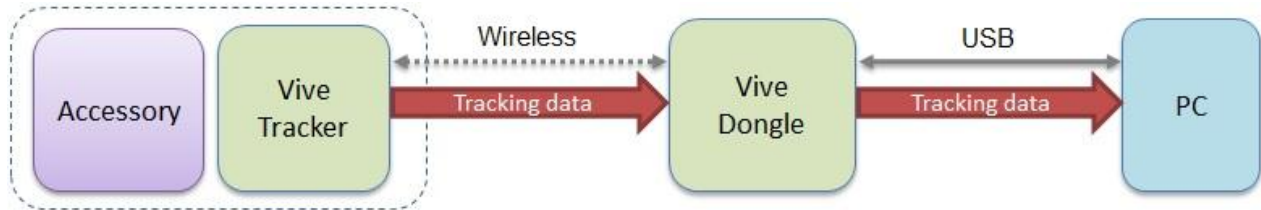


Figure: Use case 3 of Vive Tracker

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

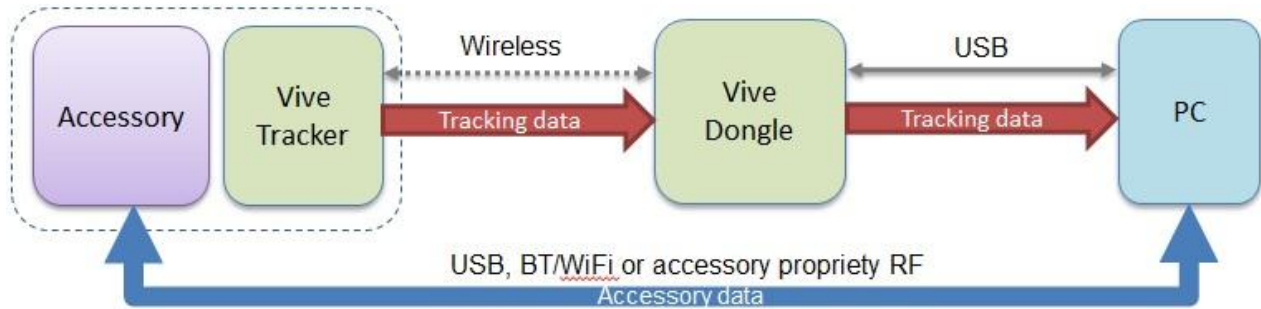


Figure: Use case 4 of Vive Tracker

**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 via the Vive Tracker. This case is similar to Use Case 3, but the accessory connects with the Vive Tracker to transfer data to/from a PC by Pogo pin or USB interface.

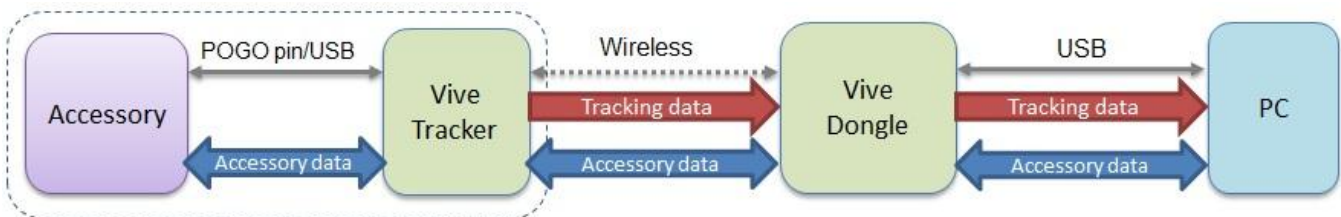
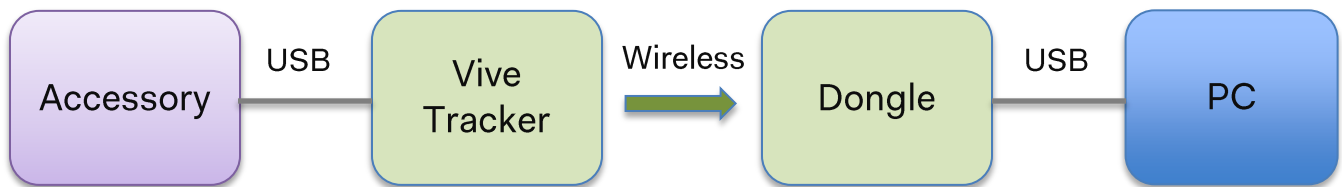


Figure: Use case 5 of Vive Tracker

## Hardware requirements

This section describes hardware requirements for accessories used with the Vive Tracker in order to enable position tracking and input of specific data for the HTC Vive VR system.

A compatible accessory may be attached to the Vive Tracker to send specific data to a PC through the USB interface of the Vive Tracker. The Vive Tracker needs to pair with the dongle to transfer data to a PC. The figure below describes the conceptual architecture.



*Figure: Conceptual Architecture of Vive Tracker*

## Interface

USB 2.0 full speed (client) from micro USB connector.

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

## Absolute Maximum Rating

Symbol	Parameter	Min	Max	Unit
$V_I$	Input voltage	- 0.5	3.6	V
$V_{ESD}$	Electrostatic discharge voltage, Human Body Model	--	4000	V

## 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	20	--	--	mA
$I_{OL}$	Low-level output current	4	--	--	mA
$I_{IH}$	High-level input current	--	0.5	10	nA
$I_{IL}$	Low-level input current	--	0.5	10	nA

## Radio frequency (RF)

To establish a stable wireless connection between the Vive Tracker and the dongle, the OTA performance of Vive Tracker cannot downgrade to more than 3dB when an accessory is attached to the Vive Tracker.

The following are recommendations for better RF performance:

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 keep at least 30mm distance away from the antenna to avoid OTA performance reduction when the accessory is attached to Vive Tracker.

The figure below illustrates the “keep out” area where only nonmetallic parts of the accessory should be inside (spherical radius=30mm and the center is antenna feed point).

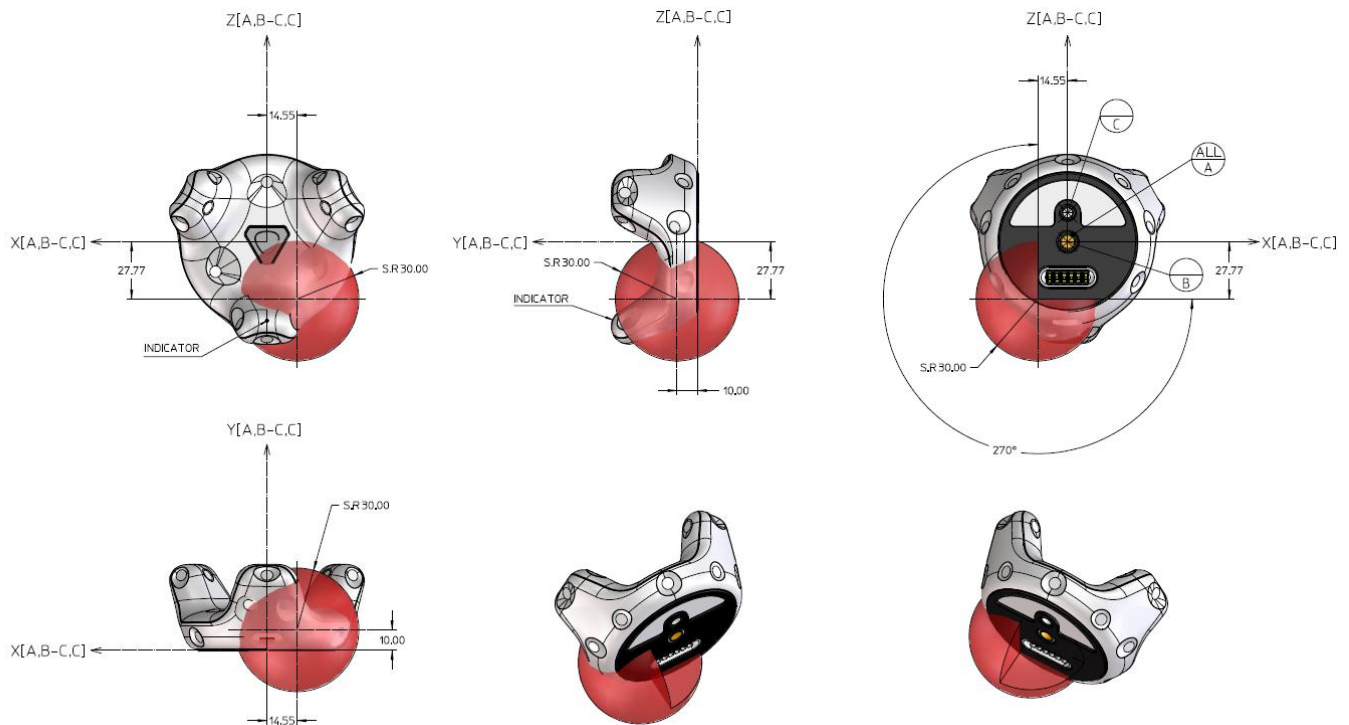


Figure: Restricted Area of Antenna



## Power

Micro USB connector	Voltage requirement	Max charging current	Max charging time
AC	5V+/-5%	1000 mA	1.5 hrs
PC		500 mA	3 hrs

Pogo pin 3	Voltage requirement	Max charging current	Max charging time
PC	5V+/-5%	500 mA	3 hrs

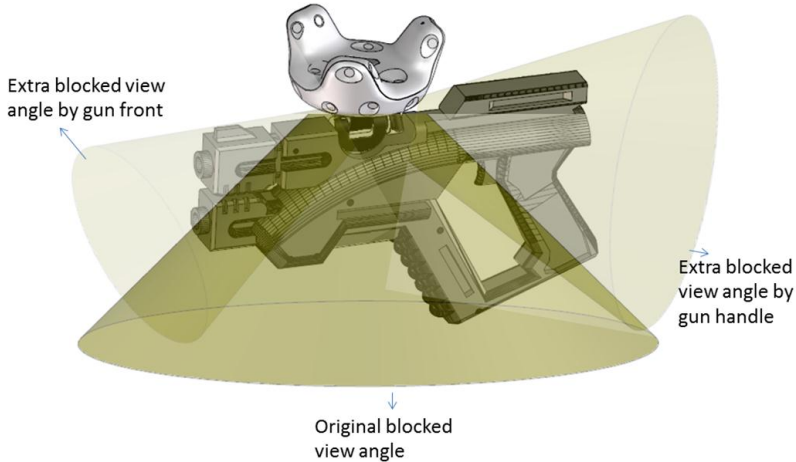
**Note**  
 AC: D+ short to D-  
 PC: D+/D- communication

*Table: Micro USB connector and Pogo pin indication*

## Optics

The field of view (FOV) of Vive Tracker is 270 degrees. Avoid placing the structure within the view angle, since it will block responses from Vive Tracker sensors when placed in that direction.

If the docking part extends beyond the recommended placing cone, extra views will be blocked.



*Figure: Docking part extends beyond recommended placing cone*

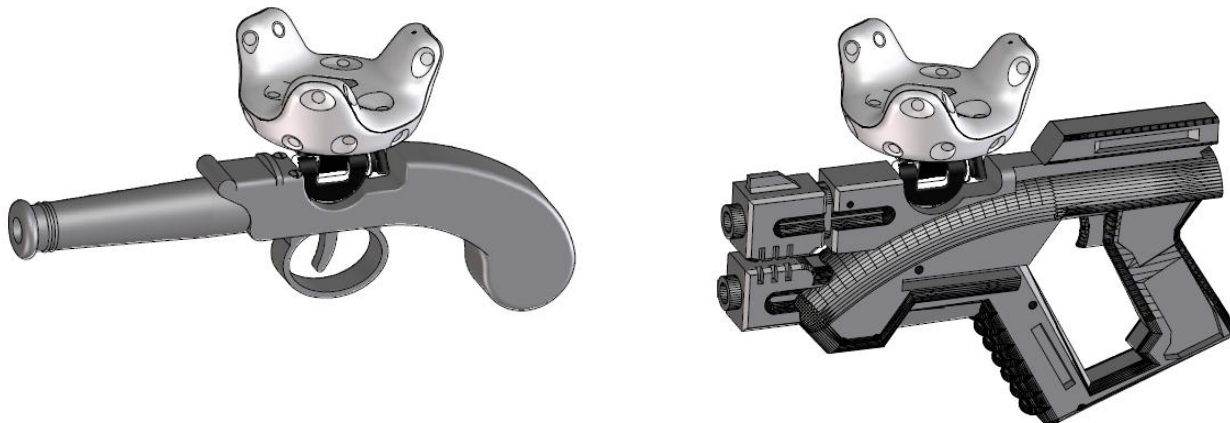
## Docking

The following are requirements for docking compatibility:

- a. The docking design of Vive Tracker follows the ISO standard (ISO 1222:2010). Furthermore, Vive Tracker has constraining features, such as the longer screw cannot fasten all the way in.
- b. The user should be able to easily attach and detach Vive Tracker with two hands. One hand holds Vive Tracker, and the other one holds the accessory.
- c. The user should not be at risk of physical harm while attaching or detaching Vive Tracker.
- d. The user should be comfortable while attaching and detaching Vive Tracker.
- e. The accessory attached with Vive Tracker should be in the shape of a physical object to avoid hitting during operating.
- f. Vive Tracker should not be blocked by the accessory and affect the tracking performance.
- g. It is strongly recommended that the accessory uses low reflection material for its outer skin to avoid reflection interference with the tracking sensors, especially if the accessory needs to be placed within the tracking sensors' FOV area.

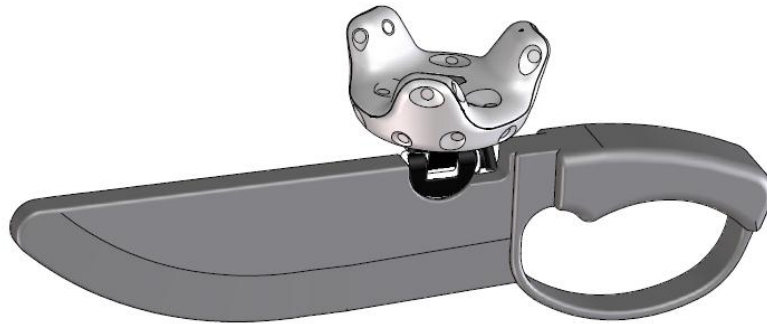
## Docking embodiments

### Gun



## Sword

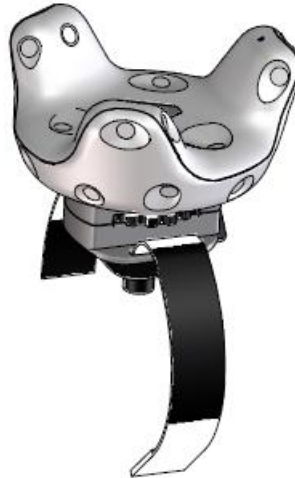
It is recommended to design the mounting mechanism close to the hand-held area, and set up the length in the VR program.



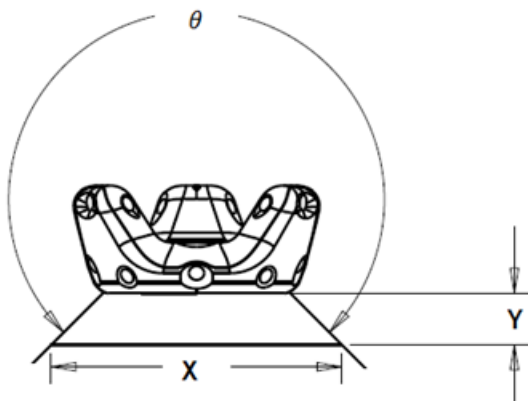
## Multi-purpose docking base

Users are able to attach the Vive Tracker to any object/surface that is intended to be tracked.

- If the object/surface is smooth and stiff, it is recommended to use stronger adhesive tape for attaching the docking base to the specific 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 specific object/surface.



Improper Vive Tracker placement may cause the accessory body to obstruct the tracking performance. The mounting distance between tracking FOV and the related accessory size is shown below:

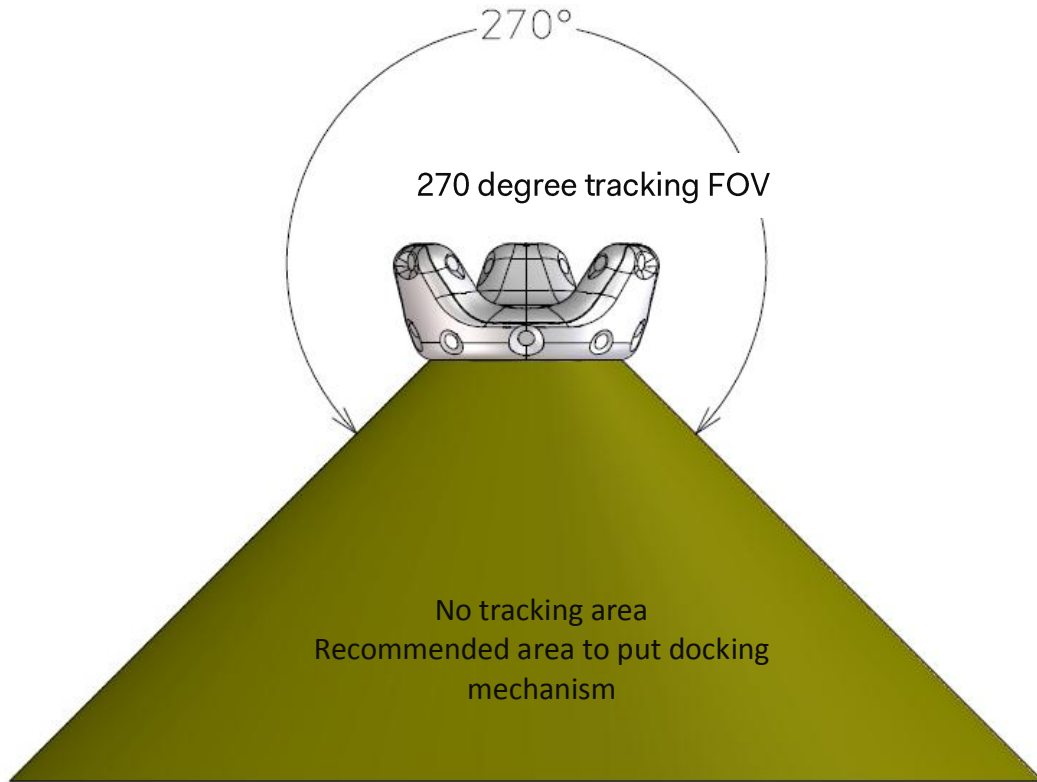


$$\theta = 180^\circ + 2 \times \tan^{-1} \frac{2Y}{X}$$

$\theta$  : Tracking FOV

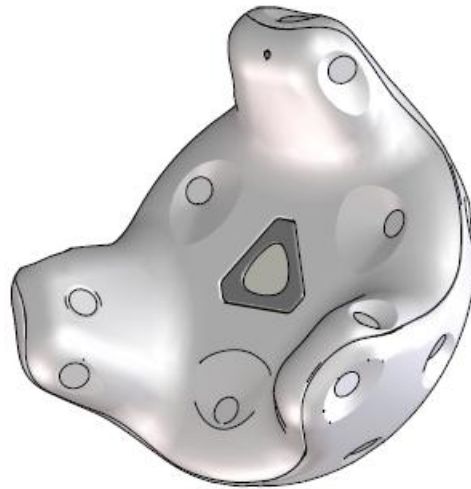
X : Docking surface width of accessory

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.



*Figure: Vive Tracker*

Apparel size

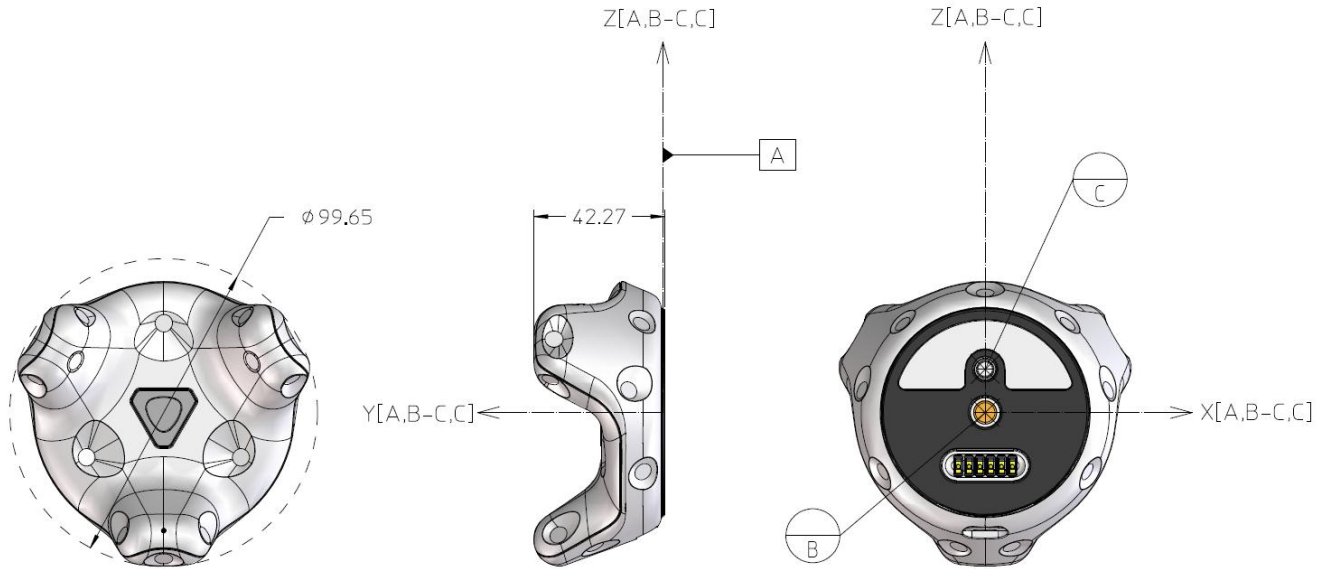


Figure: Vive Tracker with different angle

The overall size of the Vive Tracker is  $\phi 99.65\text{mm} * 42.27\text{mm (H)}$ .

## Main feature

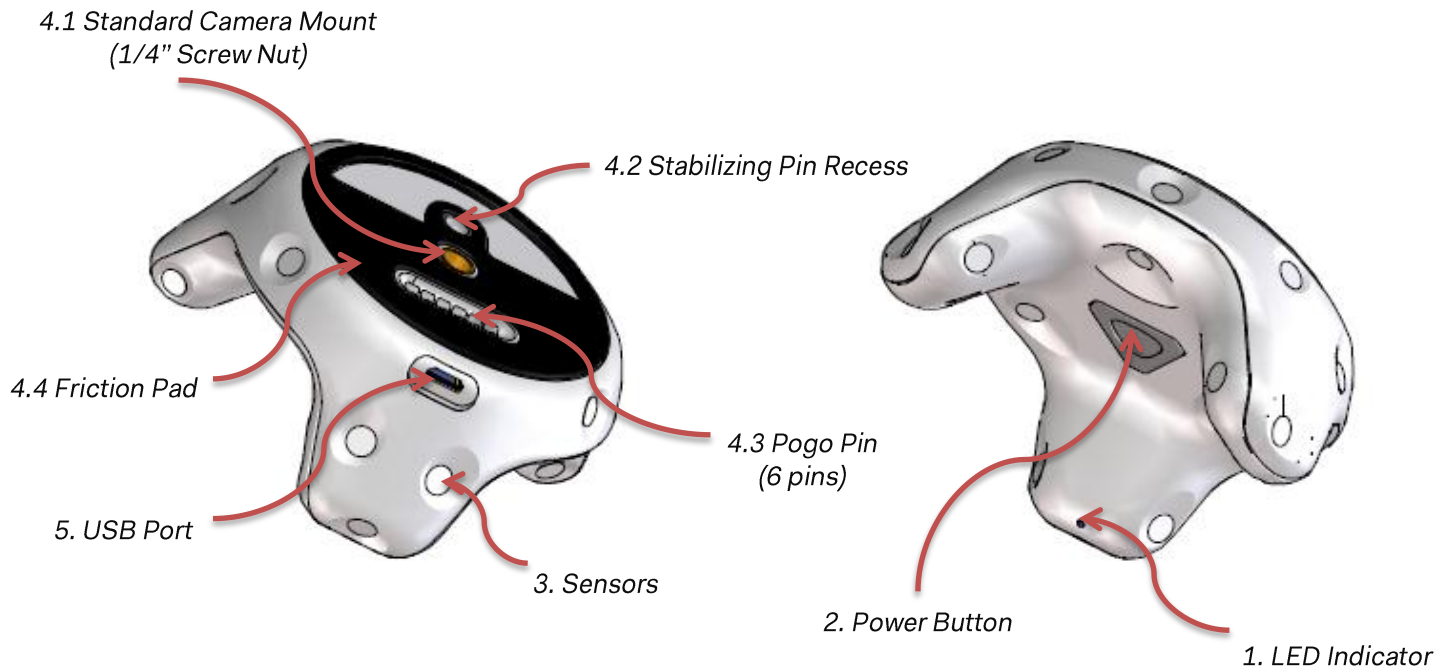


Figure: Main features

1. **LED Indicator:** Shows the status of Vive Tracker.
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. Accessory should minimize surface reflection (e.g. avoid white color surface) since it may cause faulty signal and affect performance. Anti-reflection painting is preferred.
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 tracking from rotation.
  - 4.3 Pogo pin port (spring contact-type) for optional electrical connection to the accessory.
  - 4.4 Friction pad for providing steady friction between the accessory and Vive Tracker.
5. **USB Port:** Used for electrical connection to the accessory through a micro USB cable.



## Docking mechanism

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

The following are the schematic drawings of how the accessory will mount to the Vive Tracker.

### Docking with standard tripod cradle head (w/o electric connection)

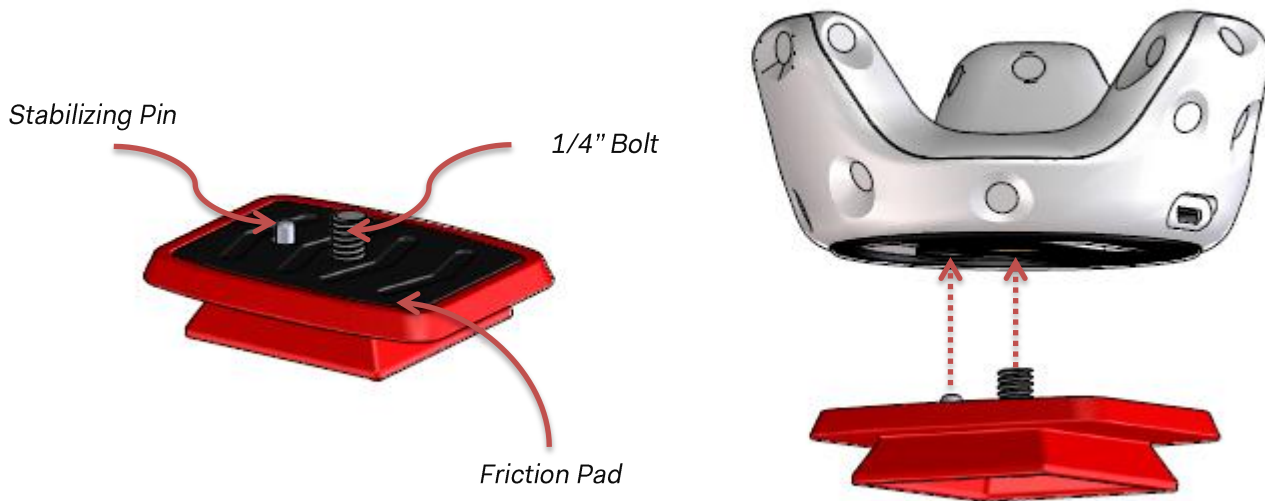


Figure: Docking with standard tripod cradle

Vive Tracker 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).

### Docking with side tightening wheel (w/ electric connection if needed)

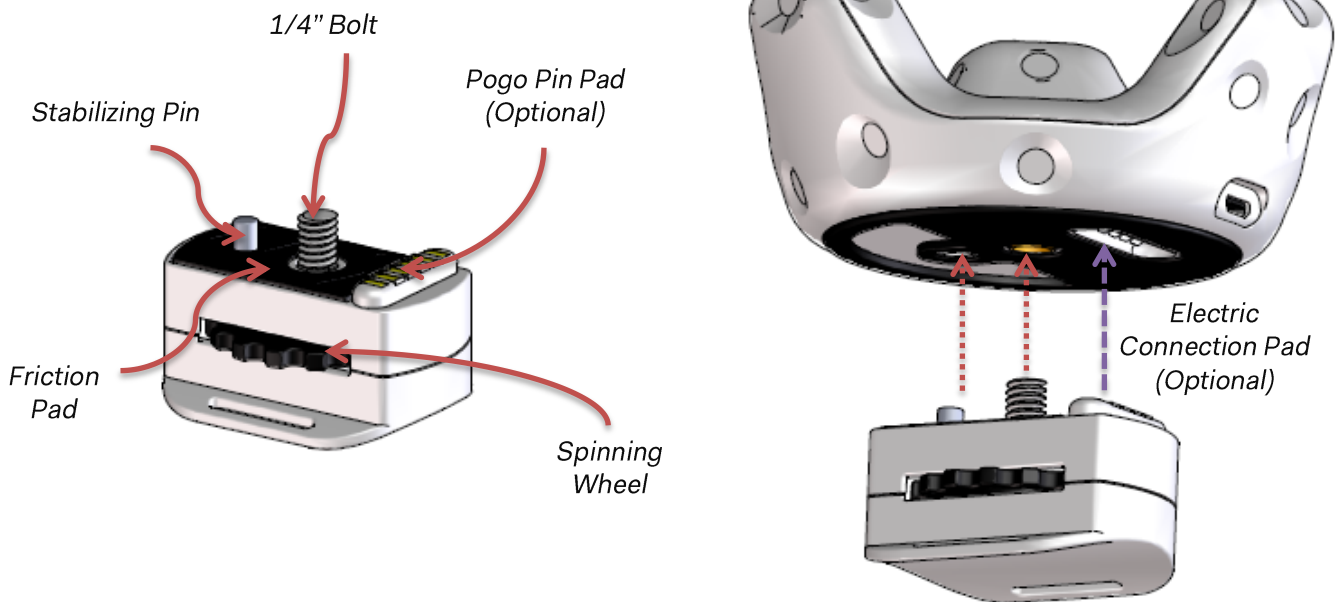


Figure: Docking with side tightening wheel

In this example, the mechanical method allows developers to tighten the docking screw through the side spinning wheel. It is recommended that the spinning wheel should have a larger diameter (25 mm or greater) for better operation.

This example is able to use the Pogo pin for extending the electric connection to somewhere else.

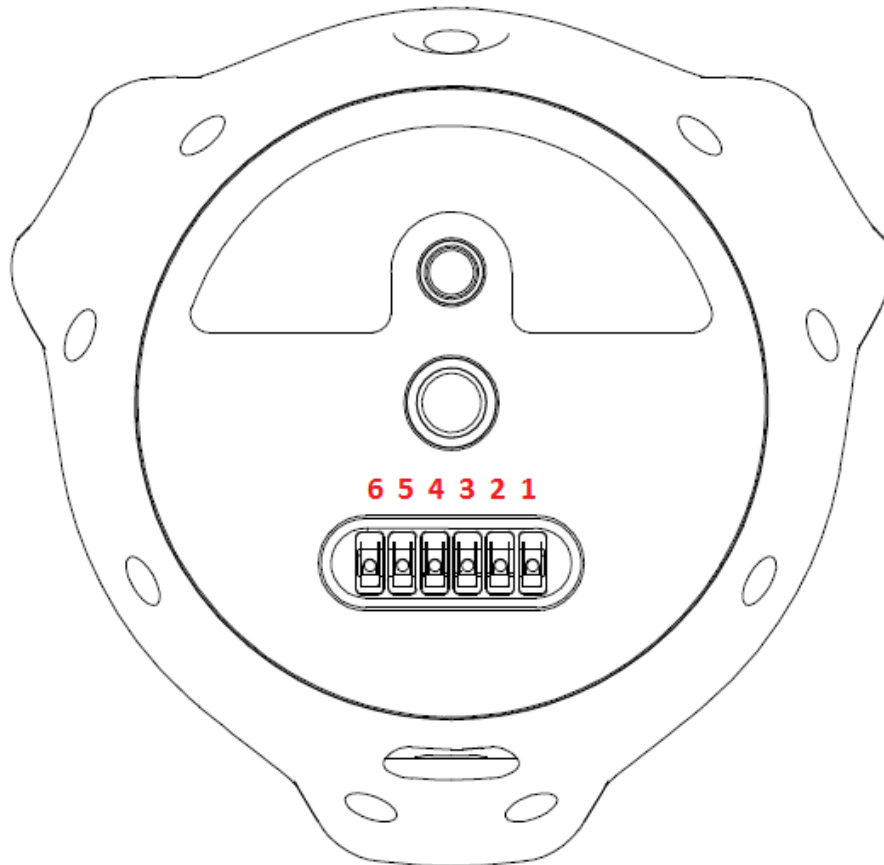
## Accessory design

Below are the different accessory mechanisms, following ISO standards:

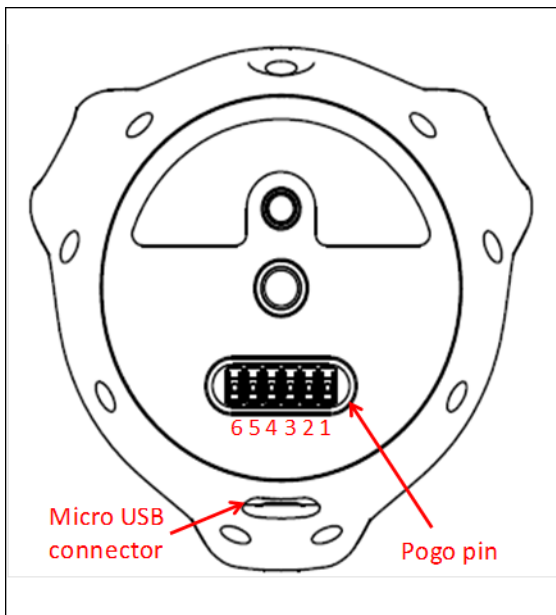
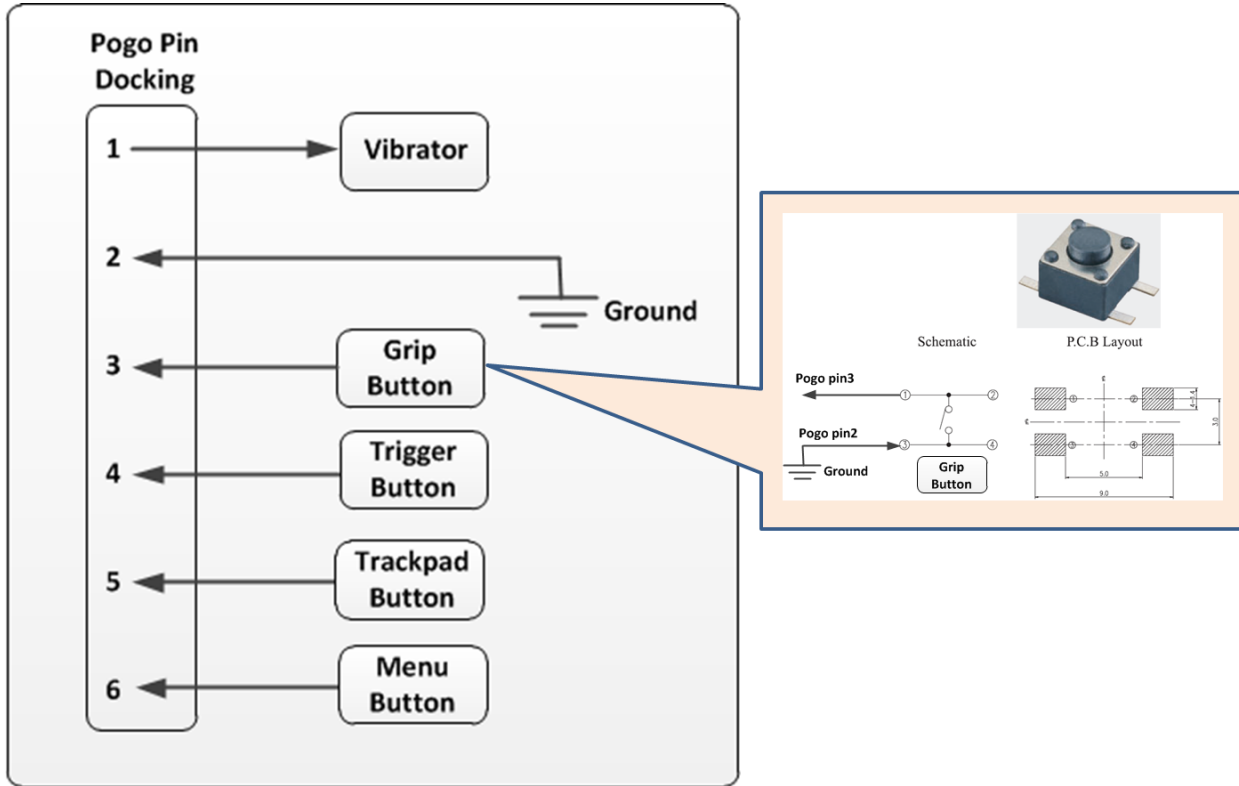
- **1/4" bolt design**  
Please refer to ISO 1222-2010, Figure 1 on page 1.
- **Stabilizing pin design**  
Vive Tracker leverages the design from ISO 1222-2010, Figure 5 on page 3. For details on dimensions and tolerances, please refer to pages 13-17. It is suggested to apply the Stabilizing Pin for better tracking performance.
- **Screw thread design**  
The screw thread type that applies to Vive Tracker is 1/4" screw with 1.27 mm pitch. For detailed information, please refer to ISO 1222-2010, pages 3-5.

## Design of Pogo Pin Pad

- a. Pin definition (Vive Tracker)

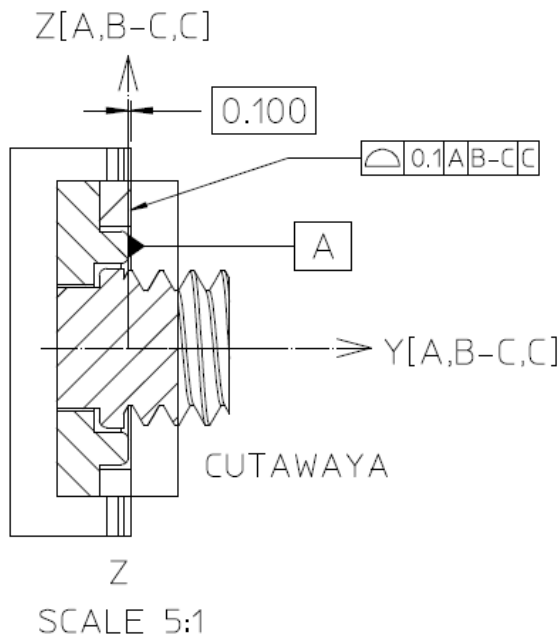
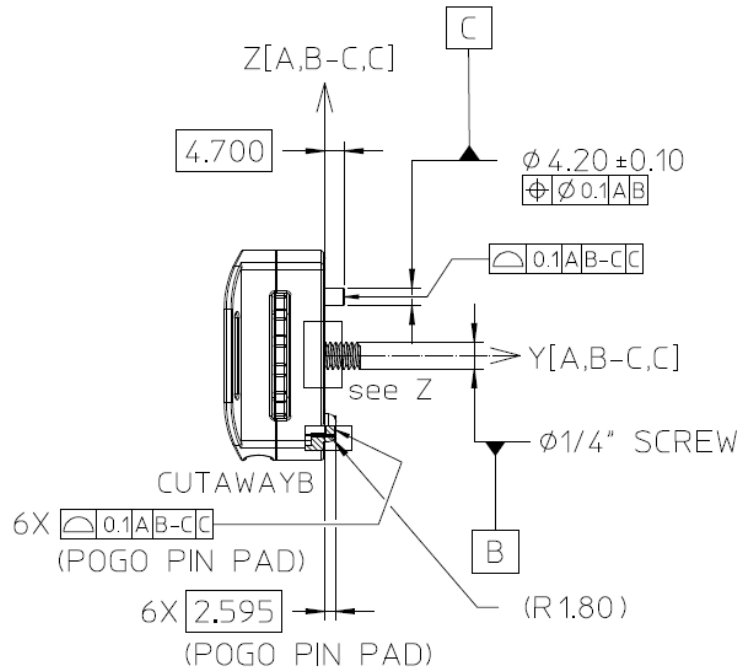


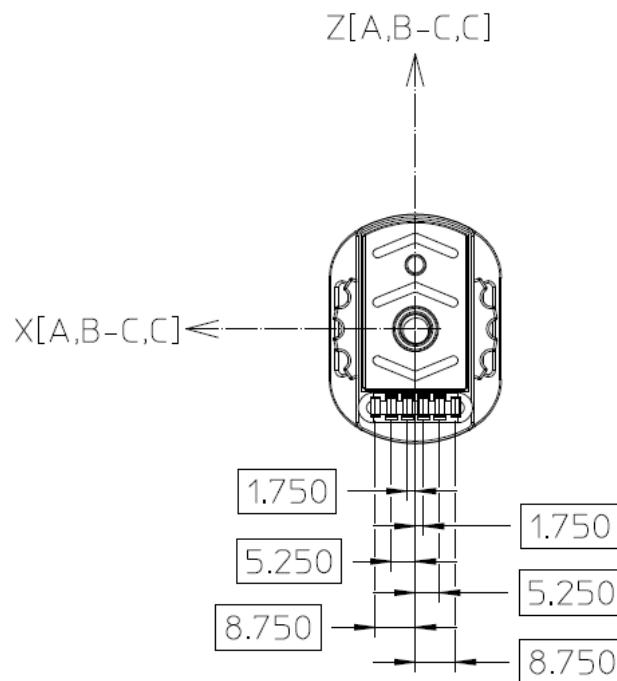
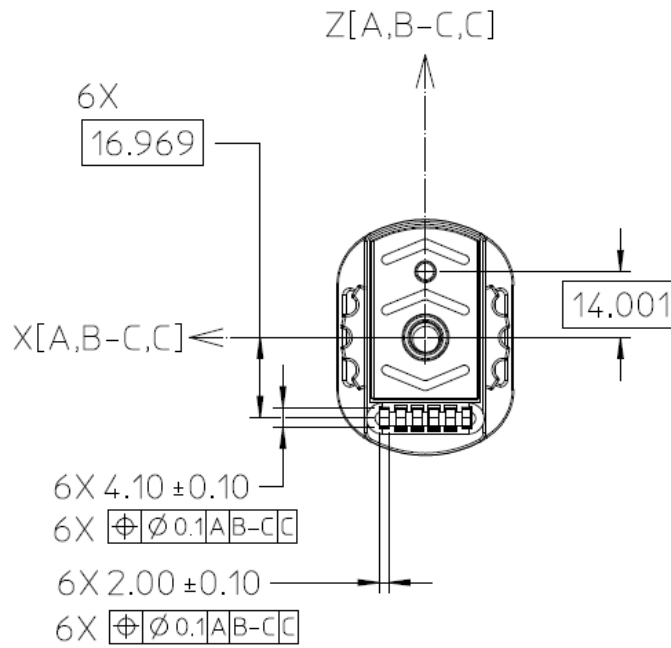
b. Pogo Pin Pad reference design  
Electrical

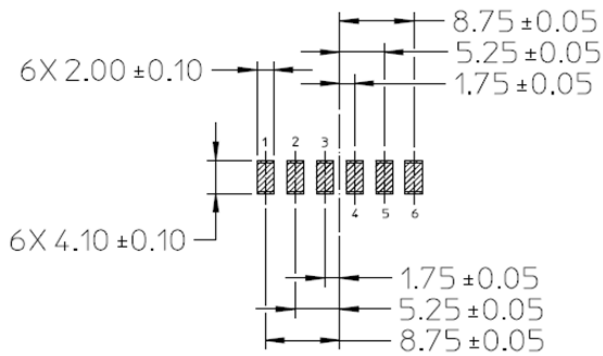
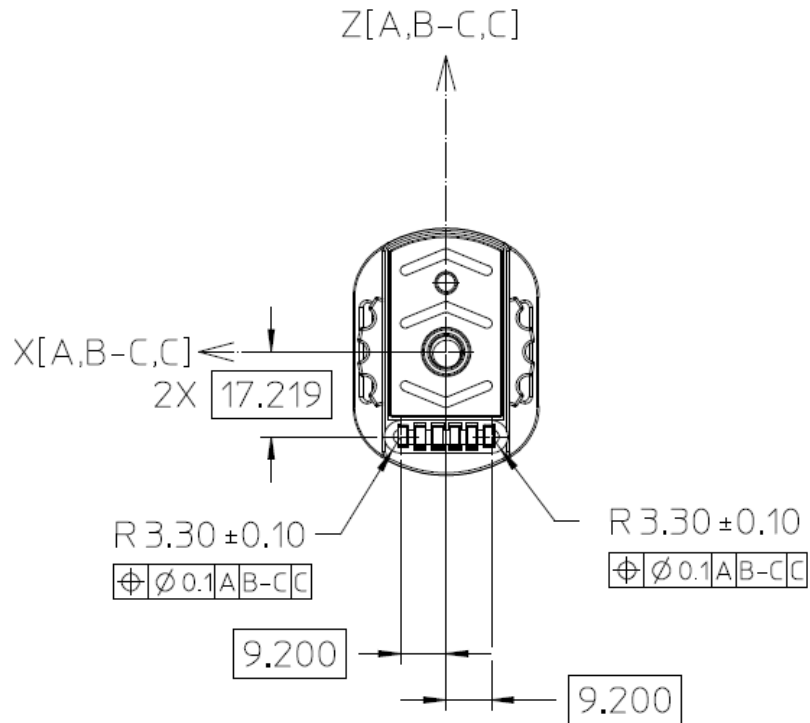


Pin no.	Type	Description
1	Digital output	General purpose output pin
2	GND	Ground
3	Digital / Power input	General purpose input pin: Internal pull up resistor to VDD, Active-low (Grip button) Power input pin
4	Digital input	General purpose input pin: Internal pull up resistor to VDD, Active-low (Trigger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







Contact Resistance :

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

Contact Current Rating :

1.8A (Min)

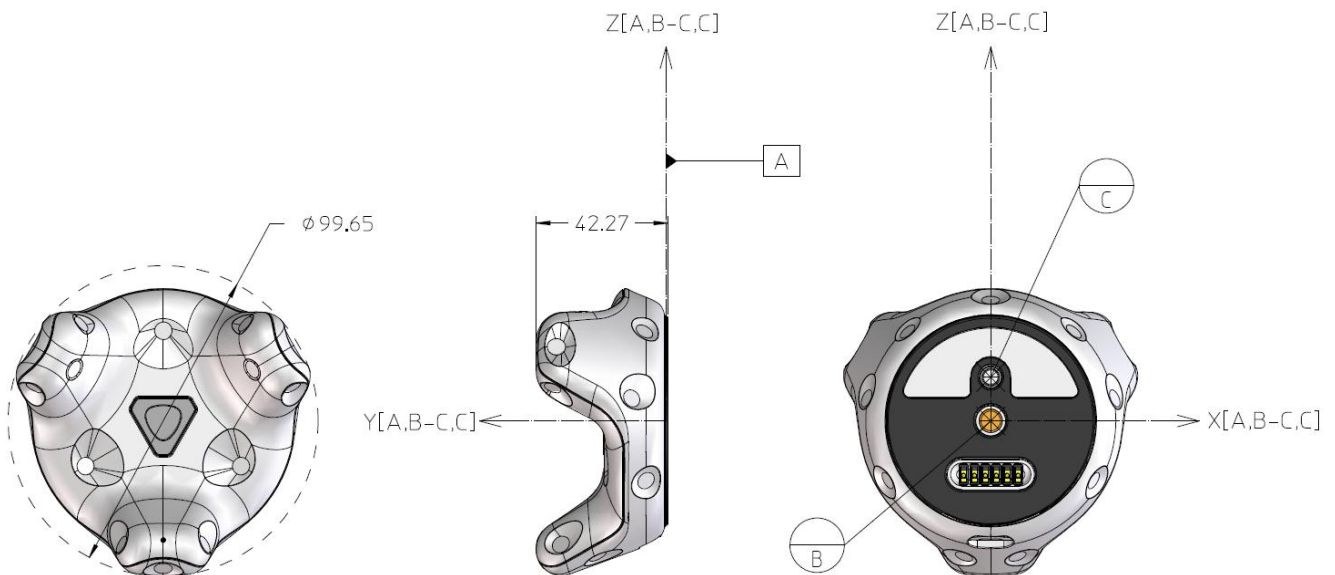


## Coordinate system

Vive Tracker uses the “Right-handed coordinate system”.

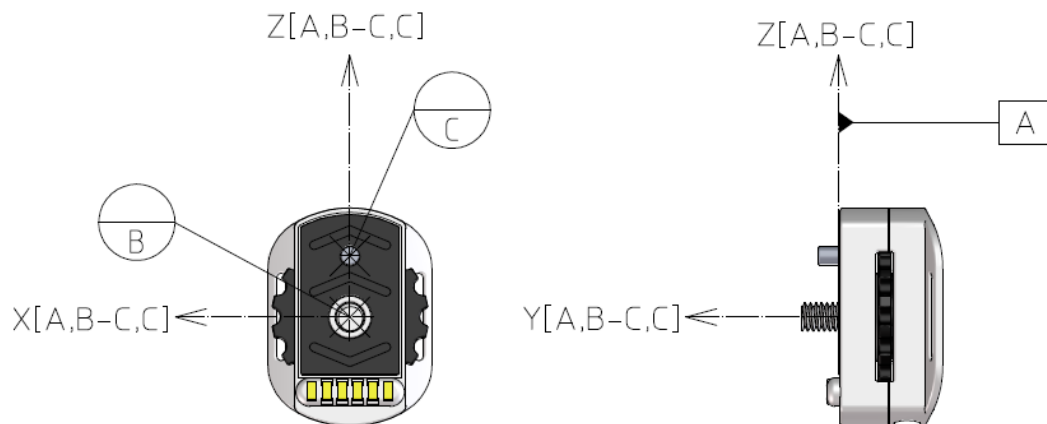
### Vive Tracker

- 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 HTC Vive Tracker.

If you are an accessory maker, you can transfer data through Vive Tracker. You may refer to the detailed data format transfer between an accessory and Vive Tracker in the [Data format](#) section.

If you are a content developer, you can refer to [Unity integration](#) and [Accessory integration](#) sections to create virtual reality content for an accessory attached with Vive Tracker.

When new firmware for Vive Tracker is released, you can upgrade the firmware using a PC tool through USB cable. You can find the steps in the [Firmware upgrade](#) section.

## System requirements

For both content developers and accessory makers:

1. To test Vive Tracker with your content or accessory, you need to have HTC Vive and the relevant hardware and software environments to run it. You can find more information on HTC Vive in [www.htcvive.com](http://www.htcvive.com)
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 previous section) or Vive Tracker (for firmware upgrade purpose). This PC should also run SteamVR for HTC Vive.

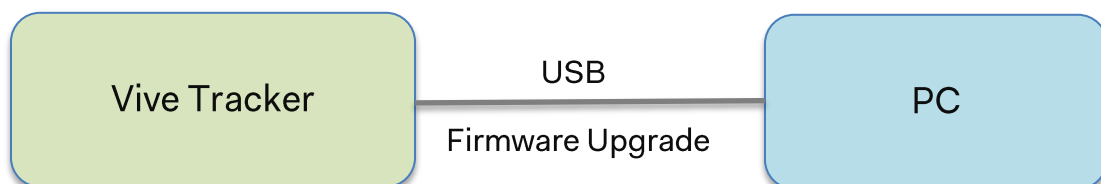
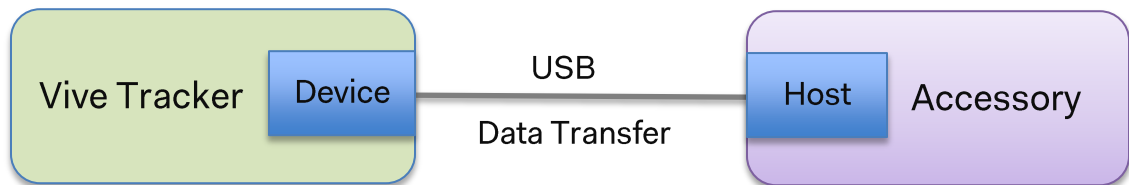


Figure: Vive Tracker and PC

**For accessory makers:**

If your accessory needs to simulate buttons of the Vive controller or transfer data to a PC through Vive Tracker, it must support the following interfaces respectively:

1. Pogo pin  
Refer to hardware requirement section for detailed information on button simulation.
2. USB full speed host and HID class. The Vive tracker will act as a USB device to transfer data to/from the accessory.



*Figure: Vive Tracker and accessory*

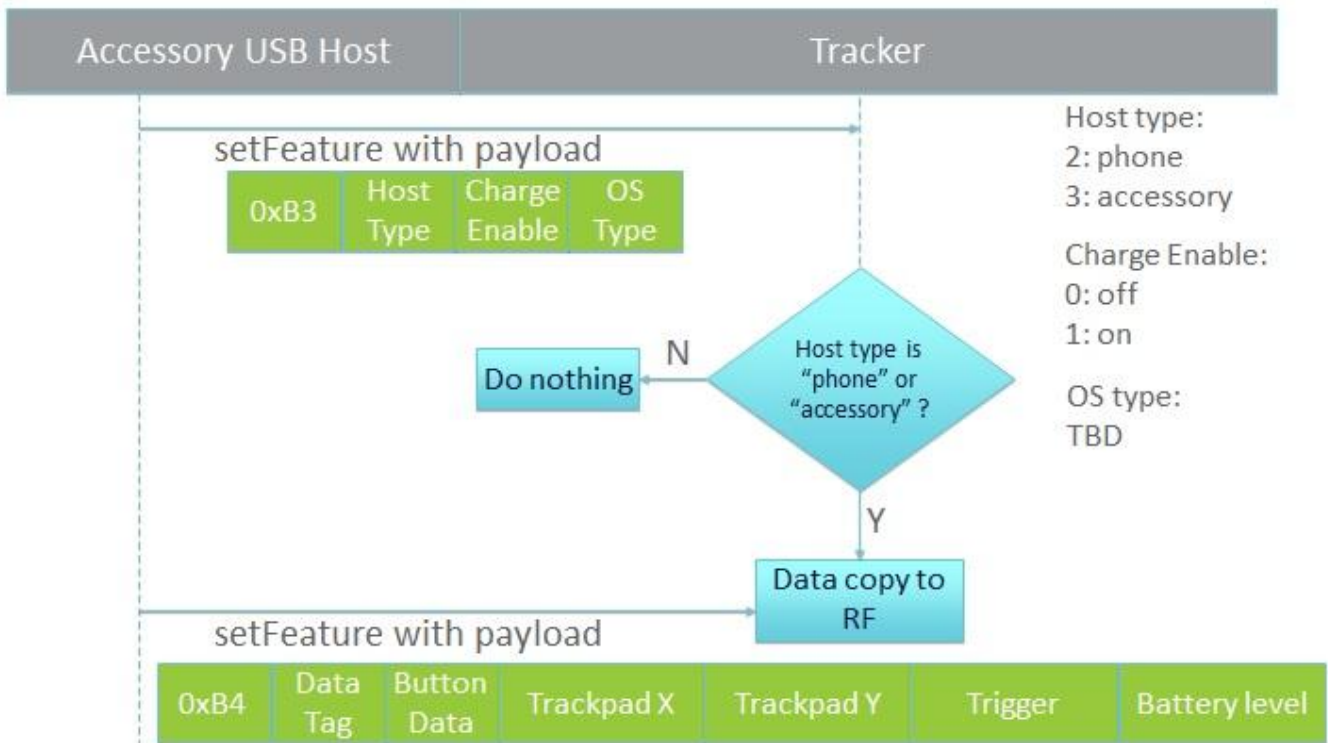
## Data format

This section describes data formats for accessory makers to transfer data between the accessory and the PC through Vive Tracker when the USB interface is used.

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

For information about the USB HID spec, refer to [USB Device Class Definition for Human Interface Devices \(HID\)](#). Vive Tracker has three USB nodes when it connects with via USB to a computer. To send the USB command to Vive Tracker, set the wIndex to 2.

Refer to the table below for the USB command flow between accessory and Vive Tracker.



### SetFeature 0xB3 data format:

Byte Index	Data	Remark
0	Host Type	2 : phone 3 : accessory
1	Charge Enable	Reserved
2	OS Type	Reserved

### SetFeature 0xB4 data format:

Byte Index	Data	Remark
0	Tag Index	Indicates the version of the data you send 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
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)*

### Sample code

Provided below are sample codes to send the setFeature command to Vive Tracker. Note that you need to reference the correct API in your system to implement 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

This section describes information on position transformation between an accessory and Vive Tracker. 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 z-axis facing the front (left-handed coordinate system), and Vive Tracker is attached in the accessory as in the example below. Rotation degree and translation distance of an accessory relevant to Vive Tracker 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, refer to guidelines related to the hardware and mechanical design.

An example of using a gun as an accessory is described in the figure below:

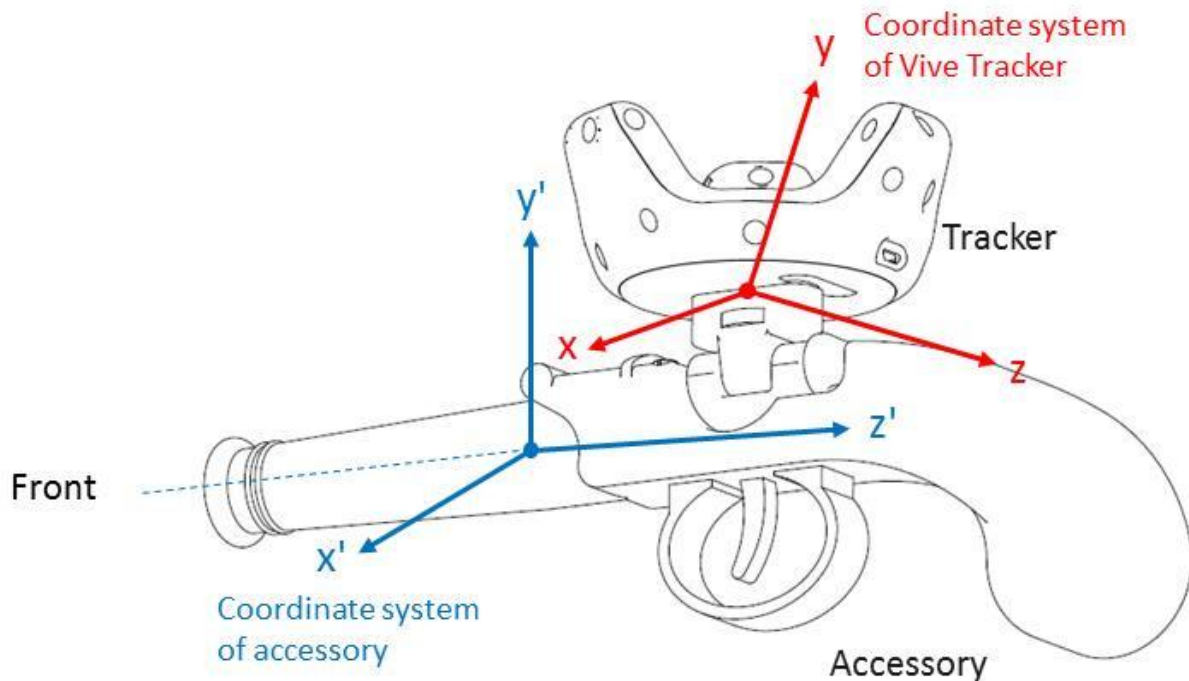


Figure: Example to integrate accessory and Vive Tracker



Pitch: Angle that rotate around x axis in degrees

Yaw: Angle that rotate around y axis in degrees

Roll: Angle that rotate around z axis in degrees

$D_x$ : 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:

```
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;

    }

}
```

Figure: Unity sample code for accessory integration (1)

Another Unity sample code shows how to transform the accessory by comparing vectors parallel to y-axis and z-axis of the Vive Tracker (AxisY\_Tracker, AxisZ\_Tracker in example below) and the accessory (AxisY\_Accessory, AxisZ\_Accessory in 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;
    }
}
```

Figure: Unity sample code for accessory integration (2)

## Unity integration

This section provides an example for content developers to enable Vive Tracker in VR content by using Unity game engine.

First, you need to make Vive Tracker detectable in SteamVR. Assume that you have two Vive controllers already, and you have plugged in the dongle into the dongle cradle to the computer's USB port. Right-click on one of the existing controller's icon and click "Pair Controller" in the pop-up menu (shown in figure below). Press the Power button on Vive Tracker for 2 seconds, and then release it to enter the pairing mode.

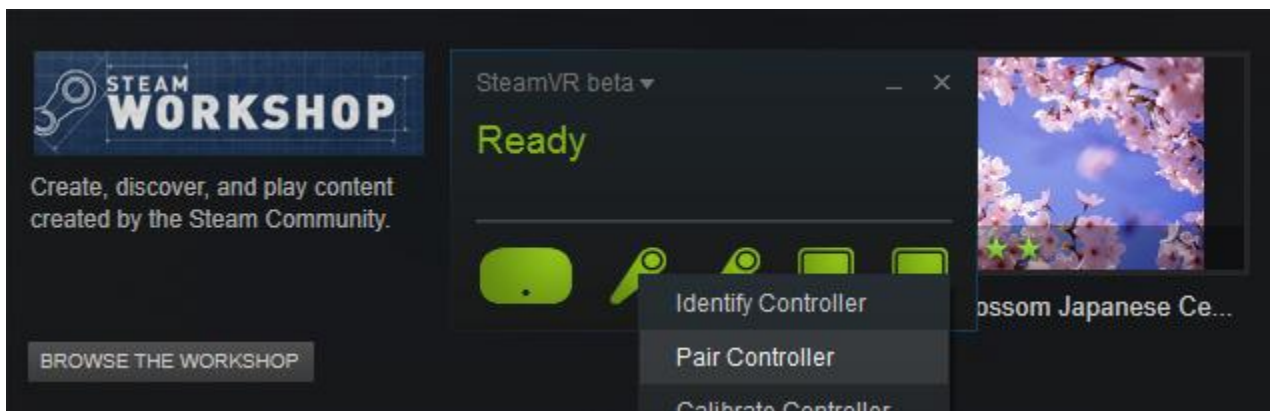


Figure: Pair Vive Tracker

After pairing is successful between the Vive Tracker and the dongle, you will see that the Vive Tracker is detected in the SteamVR UI.

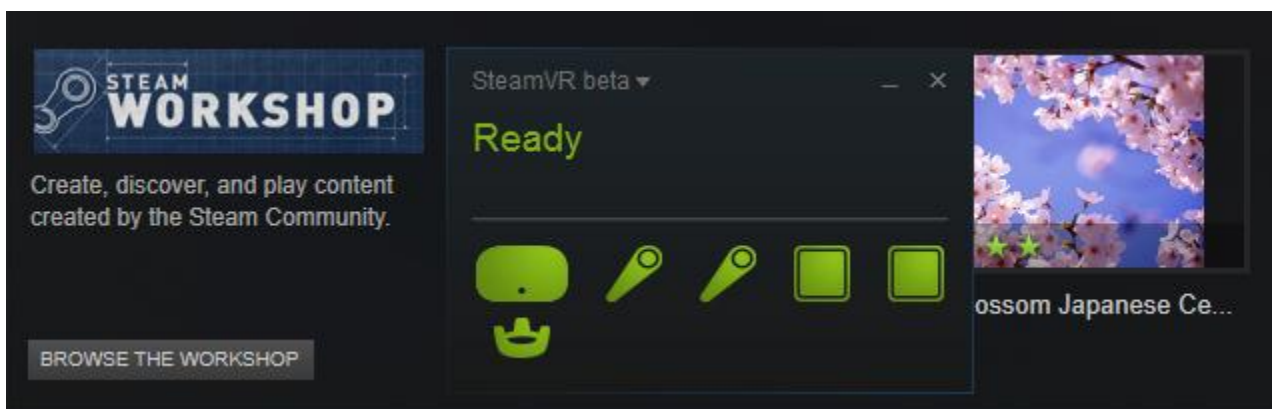


Figure: Vive Tracker is added in SteamVR

You can download Unity is 5.3.5f1 Personal from <https://store.unity.com/download?ref=personal>

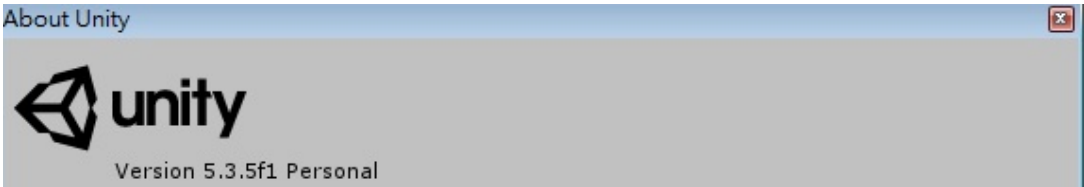


Figure: Unity Version

You need to import SteamVR Plugin into your project first. If you do not have it yet, you can download it from the Asset Store in Unity.



Figure: Unity Asset Store

In the developer version of Vive Tracker, it will use a similar approach and naming as you did to create content for the Vive controller. The steps to create content for Vive Tracker are as follows (with figures from Unity):

**Step 1:** Add “CameraRig” to Hierarchy to start creating content for SteamVR in Unity.

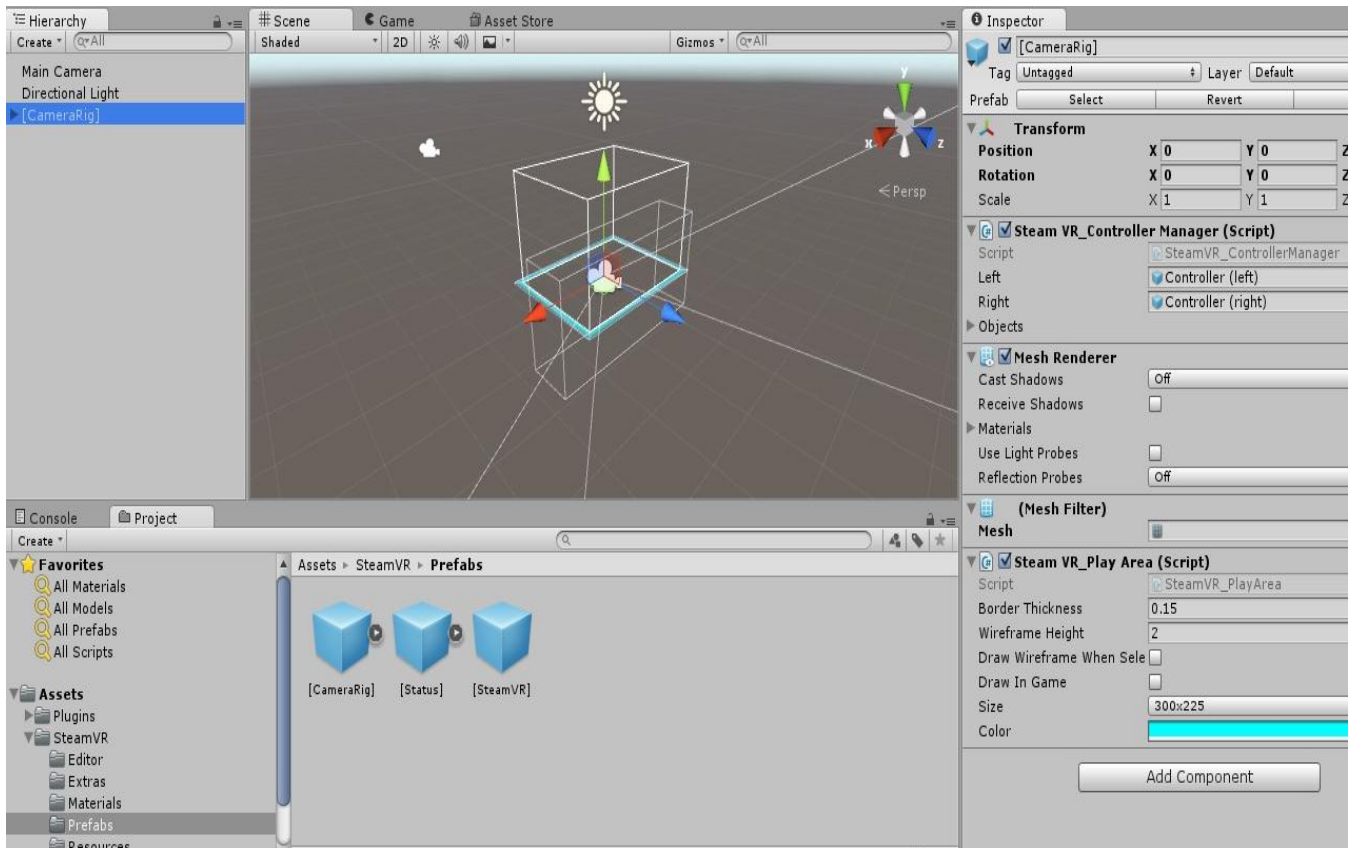


Figure: Add “CameraRig”

**Step 2: Create the 3D Object for Vive Tracker.** In this example, “Capsule” is used.

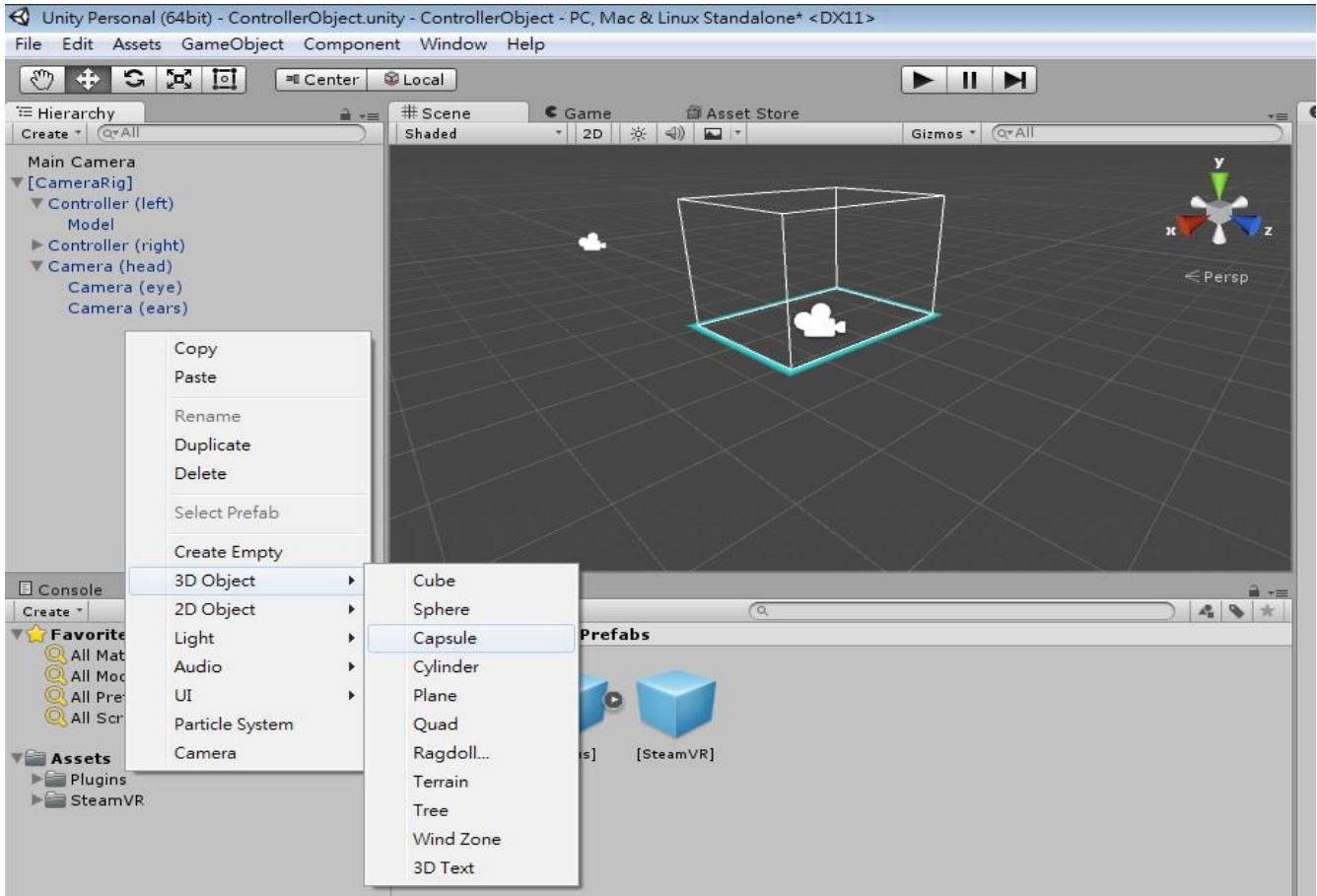


Figure: Create 3D Object



**Step 3: Add Component > SteamVR\_Tracked Object to the 3D Object “Capsule”.**

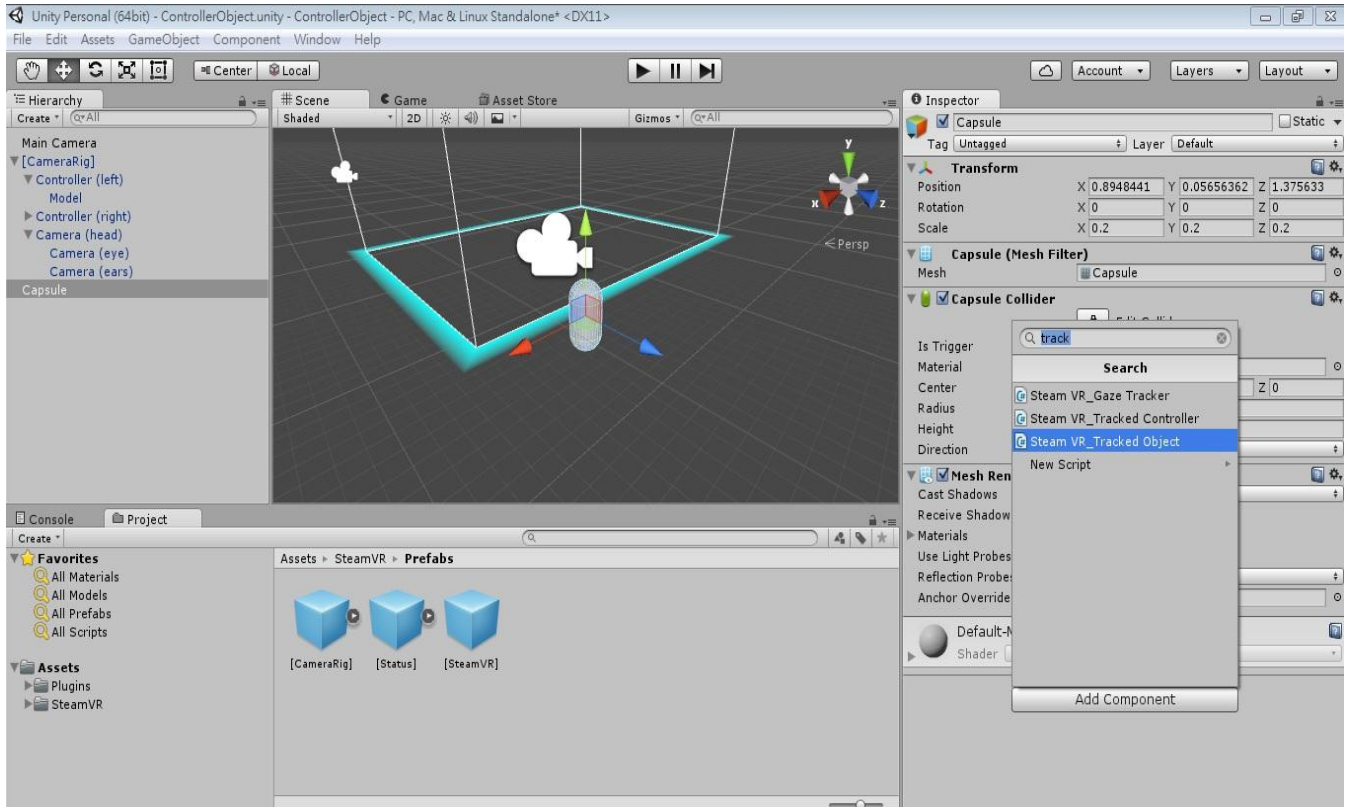


Figure: Add Component “SteamVR\_Tracked Object”



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

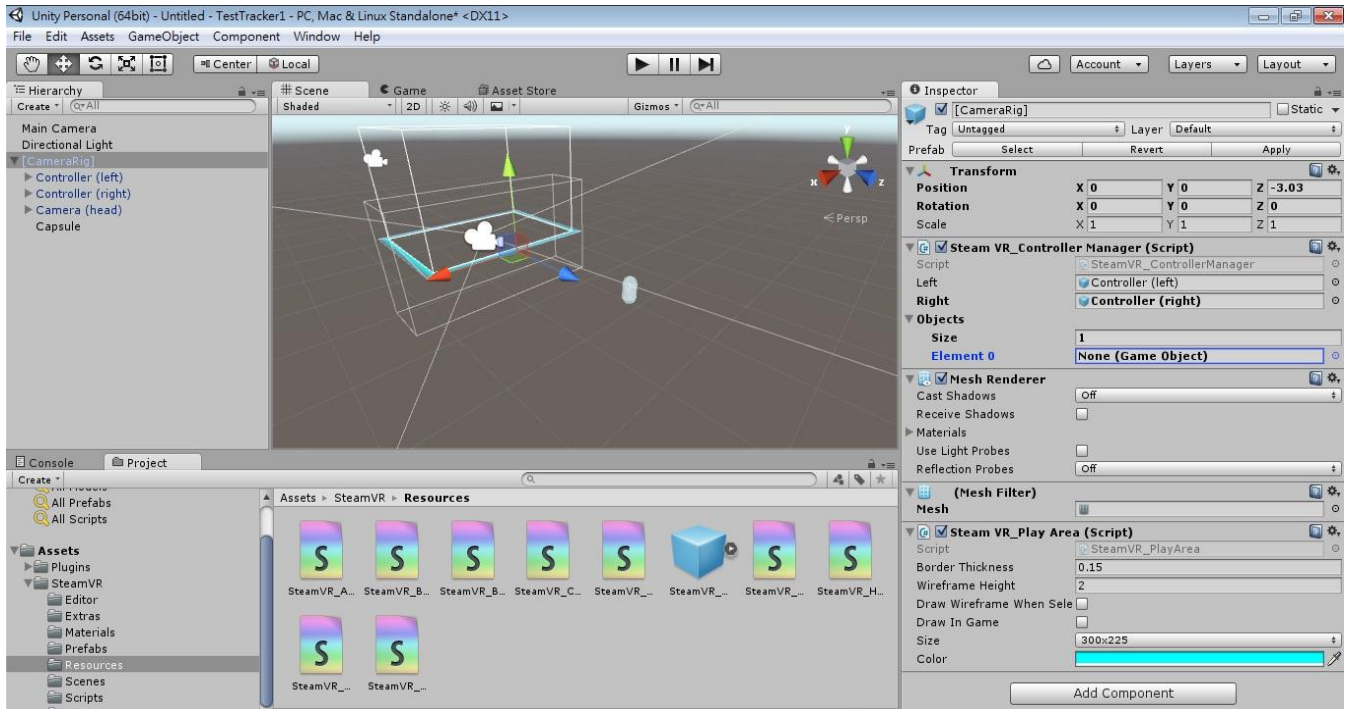


Figure: Set size of object in "SteamVR Controller Manager"

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

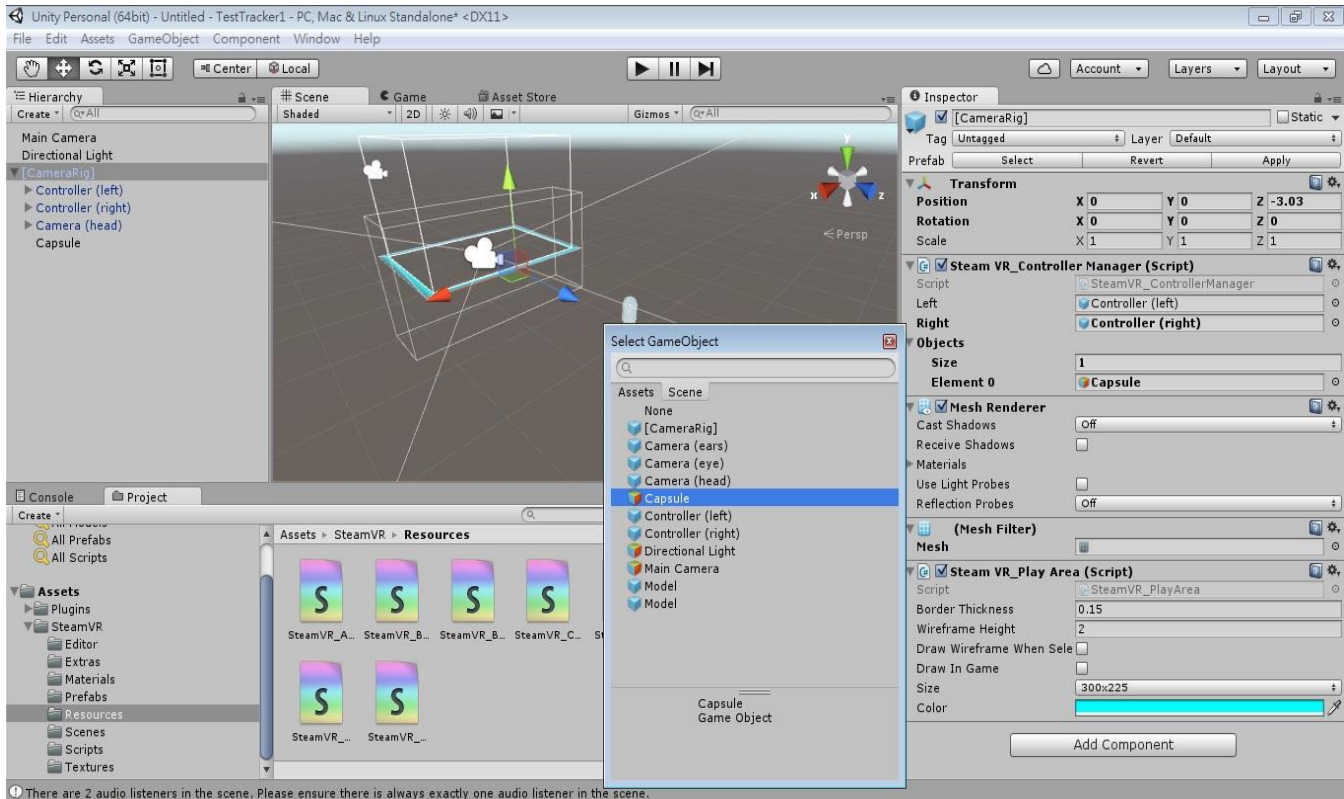


Figure: Set type of object in “SteamVR Controller Manager”

**Step 6:** After completing the steps mentioned above, press the “Run” button in Unity. When you move Vive Tracker, you will see the Capsule object is also moving in the content.

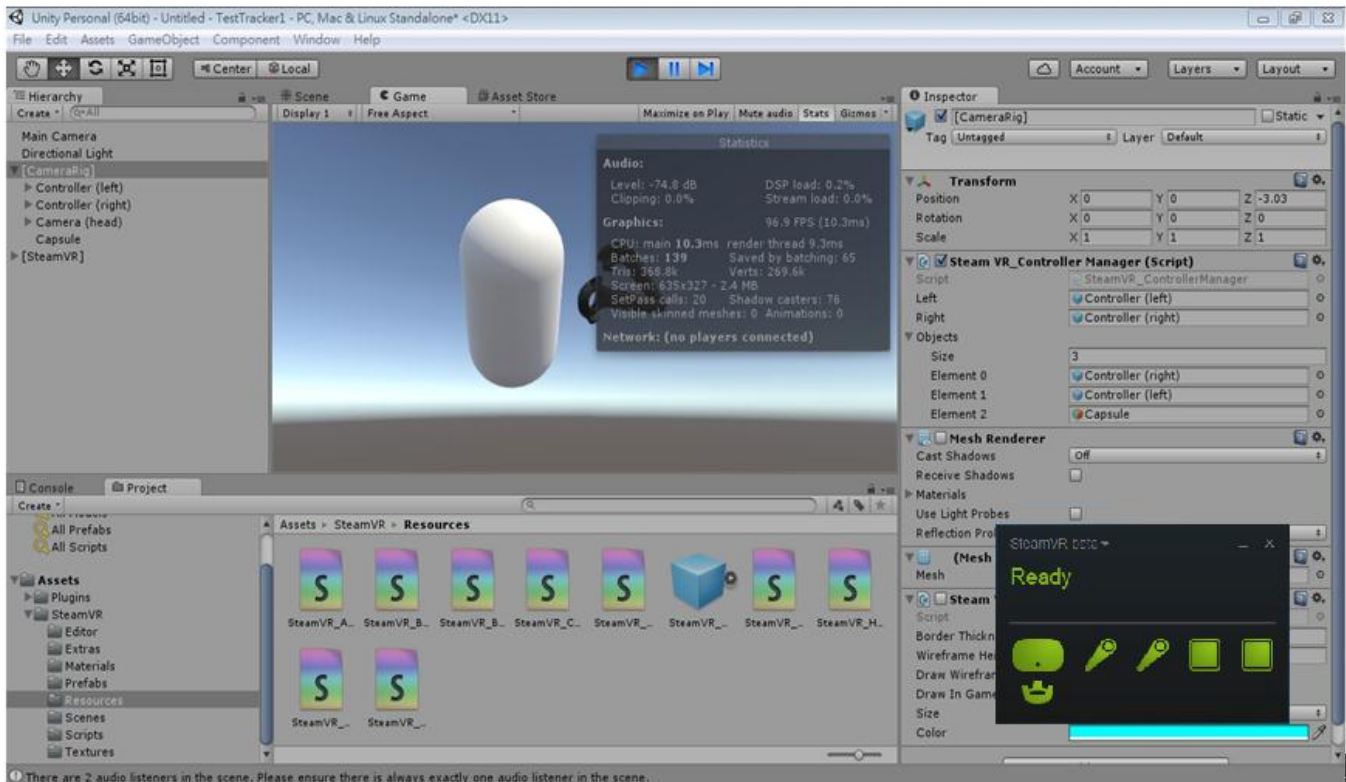


Figure: Execution of Unity

## Tracker on Unity or Unreal

If you encounter problems in enabling Vive Tracker on Unity or Unreal, refer to the following links:

- For Unity 3D developers:
  - Download link for the ViveInputUtility package: [AssetStore](#) or [GitHub](#)
  - Unity Plugin: <https://www.assetstore.unity3d.com/en/#!/content/64219>
  - Unity source code: <https://github.com/ViveSoftware/ViveInputUtility-Unity>
- For Unreal Engine 4 developers:
  - Download link for the UE4ViveTrackerSupport source code modification: [GitHub](#). ([UE4 / GitHub subscription](#) is required to access the UE4ViveTrackerSupport source code modification.)
  - Unreal source code: <https://github.com/ViveSW/UE4ViveTrackerSupport>

## Firmware upgrade

In the developer version (DVT) of Vive Tracker, the ROM files for firmware upgrade will be provided by HTC, utilizing the existing program “lighthouse\_watchman\_update.exe” of SteamVR located in the folder “SteamVR\tools\lighthouse\bin\win32\”. It should be available in your environment after the installation of the HTC Vive VR system.

For devices in DVT stage, please follow these steps to upgrade the firmware with the upgradeFirmware.zip file.

\*Note: Check the device number on the power key of your Vive Tracker. If the device number starts with D, it is in DVT stage.

1. If a Vive controller is connected to your computer via USB, unplug it first.
2. Connect one Vive Tracker with your computer using the USB cable.
3. Unzip the file upgradeFirmware.zip.
4. Execute the file upgradeFirmware.exe.
5. When done, disconnect the USB cable to reboot device.
6. To verify tracking, check that you see the Vive Tracker icon on SteamVR beta.

For devices post-DVT stage, follow the SteamVR notification to update the firmware. You may also do these steps to upgrade the Vive Tracker firmware:

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 via USB, unplug it first.
3. Connect one Vive Tracker with your computer using the USB cable.
4. Execute the following commands.

a. Update MCU's firmware:

```
lighthouse_watchman_update -mnt tracker_mcu_filename.bin
```

b. Update FPGA's firmware:

```
lighthouse_watchman_update -fnt tracker_fpga_filename.bin
```

c. Update RF's firmware:

```
lighthouse_watchman_update -rnt tracker_rf_filename.bin
```

## FAQ

Type	No.	Question	Answer
Common	1	How do I connect the dongle to my computer?	Connect one end of the supplied USB cable to the dongle cradle, and then attach the dongle to the cradle. Connect the other end of the USB cable to your computer.
	2	How do I charge Vive Tracker?	<p>While in VR, you can check the battery level of Vive Tracker when no apps are running, or when the System Dashboard is up. When the battery is low, the indicator will show a single red dot.</p> <p>Make sure to use the USB cable that's in the box. Connect the USB cable to the power adapter that came with your Vive controllers, and then plug the power adapter to a power outlet to charge Vive Tracker. You can also connect Vive Tracker to a computer's USB port to charge it.</p> <p>When Vive Tracker is fully charged, its status light either shows white if it is off or green if it is turned on.</p>
	3	How do I pair the dongle with Vive Tracker?	<ol style="list-style-type: none"> <li>1. Turn on the headset and pair the 2 controllers.</li> <li>2. Connect the dongle to your computer.</li> <li>3. Turn on Vive Tracker to start the pairing process automatically.</li> </ol> <p><b>Warning:</b> If Vive Tracker cannot be automatically paired the first time, the dongle may pair with another equipment.</p> <p>To manually pair Vive Tracker, open the SteamVR app, tap the down arrow, and then select <b>Devices &gt; Pair Controller</b>. Follow the on-screen instructions to complete the process.</p>
	4	What does the status light on Vive Tracker mean?	<p>The status light shows:</p> <ul style="list-style-type: none"> <li>• Green when Vive Tracker is in normal mode</li> <li>• Blinking red when battery is low</li> <li>• Blinking blue when Vive Tracker is pairing with the headset or dongle</li> <li>• Blue when Vive Tracker is connecting with the headset or dongle</li> <li>• Orange when charging</li> </ul>

5	How can I update Vive Tracker?	<ol style="list-style-type: none"> <li>From your computer, open the SteamVR app.</li> <li>Using the supplied USB cable, connect the Vive Tracker to one of your computer's USB ports. The firmware update will start automatically once tracker is detected by SteamVR.  <b>Warning:</b> Do not unplug the micro-USB cable any time before the firmware update is complete. Doing so could result in a firmware error.</li> <li>When the update is complete, click <b>Done</b>.</li> </ol>
6	Why does Vive Tracker automatically turn off?	<p>When Vive Tracker turns off, some of the possible reasons might be:</p> <ul style="list-style-type: none"> <li>The battery is drained</li> <li>Pairing has timed out after being idle for more than 30 seconds</li> <li>There was no user movement for 5 minutes</li> </ul>
7	Can I use more than one Vive Tracker?	Multiple Vive Tracker hardware can be used, but limitations apply.
8	Why can't I see Vive Tracker on SteamVR?	Check that your SteamVR app is the latest version.
9	How do I reset Vive Tracker?	To reset Vive Tracker, connect it to your computer using the supplied USB cable, and then press and hold the Power button for 10 seconds.
10	How many Tracker can be adapted?	The maximum number of tracking objects can be adopted is 11 pcs of Vive Tracker plus 2 pcs of Vive controller.

Type	No.	Question	Answer
SDK	1	How do identify whether Vive Tracker or the controller is being tracked?	You can use the class type of SteamVR SDK to identify if Vive Tracker or the controller is currently being tracked.
	2	Does Vive Tracker have a different coordination system compared to the controllers?	Yes. Content developers need to apply different coordination settings based on the position where the Vive Tracker is mounted.
	3	How can I replace the controller with Vive Tracker if I already created content meant for the controller?	Content developers need to apply different coordination settings based on whether Vive Tracker is mounted on the same position as the controller. If its position is different, content developers need to recalibrate for Vive Tracker.
	4	What's the equivalent of Vive Tracker's hardware button to the controller?	The hardware button of Vive Tracker maps to the System button of the controller.
	5	How can I simulate the button event when I connect Vive Tracker to a computer through USB?	Check the "Data format" section in the developer guidelines, and follow the SetFeature 0xB4 format to fill your simulated data in "Button byte" data.
	6	How can I simulate the controller trackpad when I connect Vive Tracker to a computer through USB?	Check the "Data format" section in the developer guidelines, and follow the SetFeature 0xB4 format to fill your simulated data in "Pad X" and "Pad Y" byte data.
	7	Why is tracking lost when I connected Vive Tracker to a USB cable?	When you connect Vive Tracker to your computer using a USB cable, it enters data sending mode. To resume tracking while connected to a USB port, follow the developer guidelines to send the SetFeature 0xB3 command to fill Host type byte.
	8	Can we implement our own data into Vive Tracker's data stream?	Vive Tracker only supports data that's compatible with the controllers.

Type	No.	Question	Answer
Hardware	1	Can I customize the POGO pins?	Yes, you can. For information on how to customize the POGO pins, contact <a href="http://www.vive.com/us/support/contactus">www.vive.com/us/support/contactus</a> .
	2	What's in the downloadable 3D CAD file for the Vive Tracker?	The 1.77KB ZIP file contains the .IGS and .STP for Vive Tracker. It's not a requirement to use the files, but it can help if you have 3D modeling software.