HTC Vive Tracker Developer Guidelines

Ver. 1.3

Version Control

Version Number	Version Date	Version Reason
1.0	2016.09.26	Initial version
1.1	2016.12.05	1. Use case image revised. 2. Pogo pin rearranged. 3. USB cable connection method deleted. 4. Data format revised.
1.2	2017.01.09	1. SteamVR monitor images revised.
1.3	2017.01.19	1. Design of Pogo Pin Pad revised. 2. Firmware upgrade revised.

Contents

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.

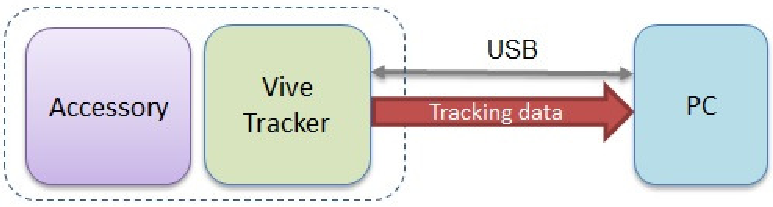


Figure: Use case 1 of Vive Tracker

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.

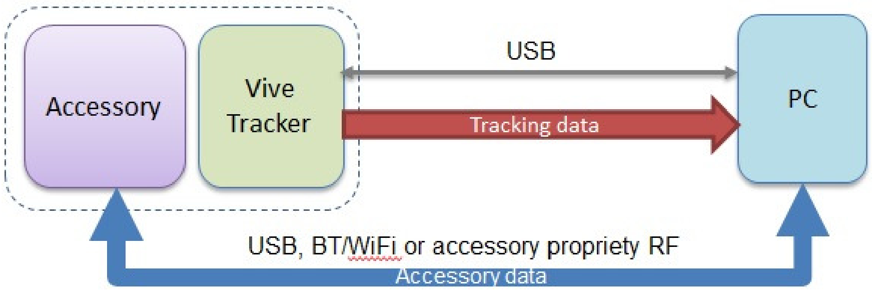


Figure: Use case 2 of Vive Tracker

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.

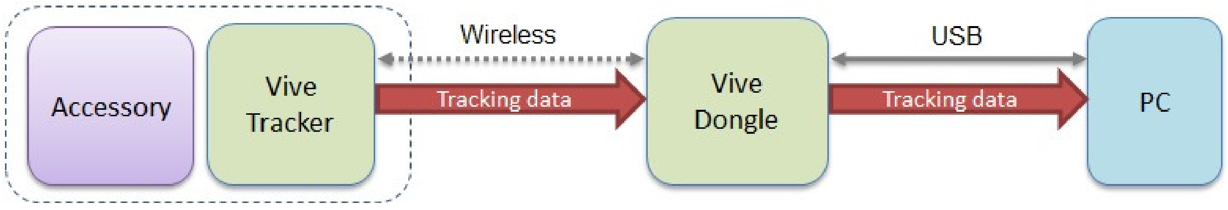


Figure: Use case 3 of Vive Tracker

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.

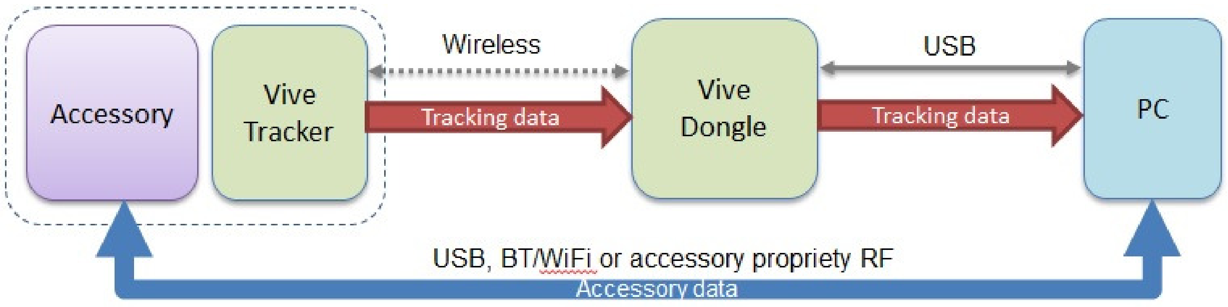


Figure: Use case 4 of Vive Tracker

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.

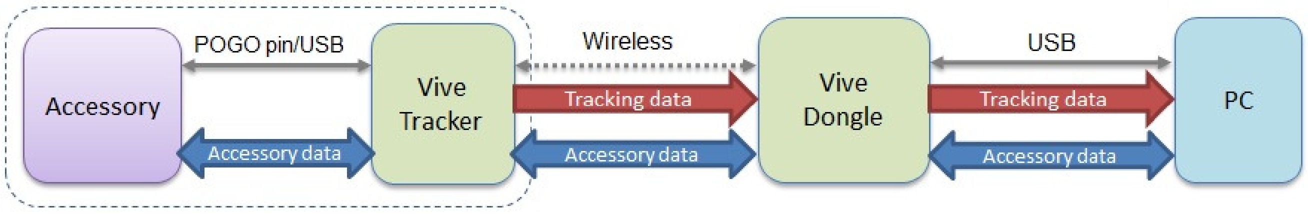


Figure: Use case 5 of Vive Tracker

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.

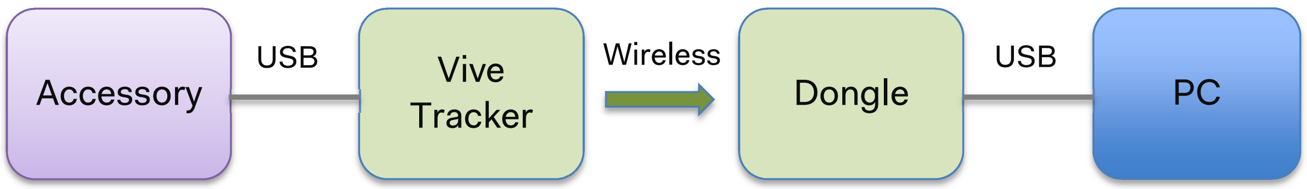
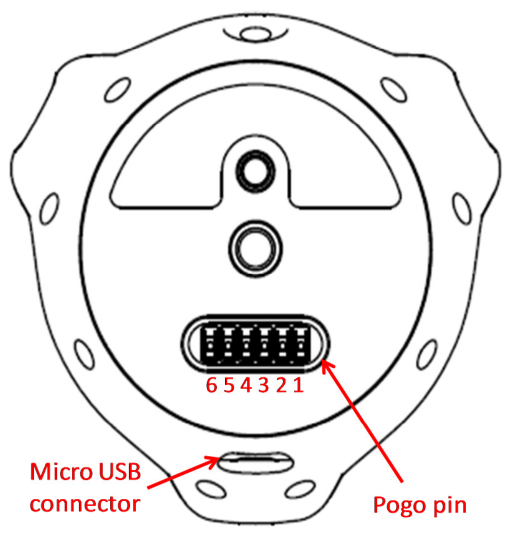


Figure: Conceptual Architecture of Vive Tracker

Interface

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



Pin no.	Type	Description
1	Digital output	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)

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 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).

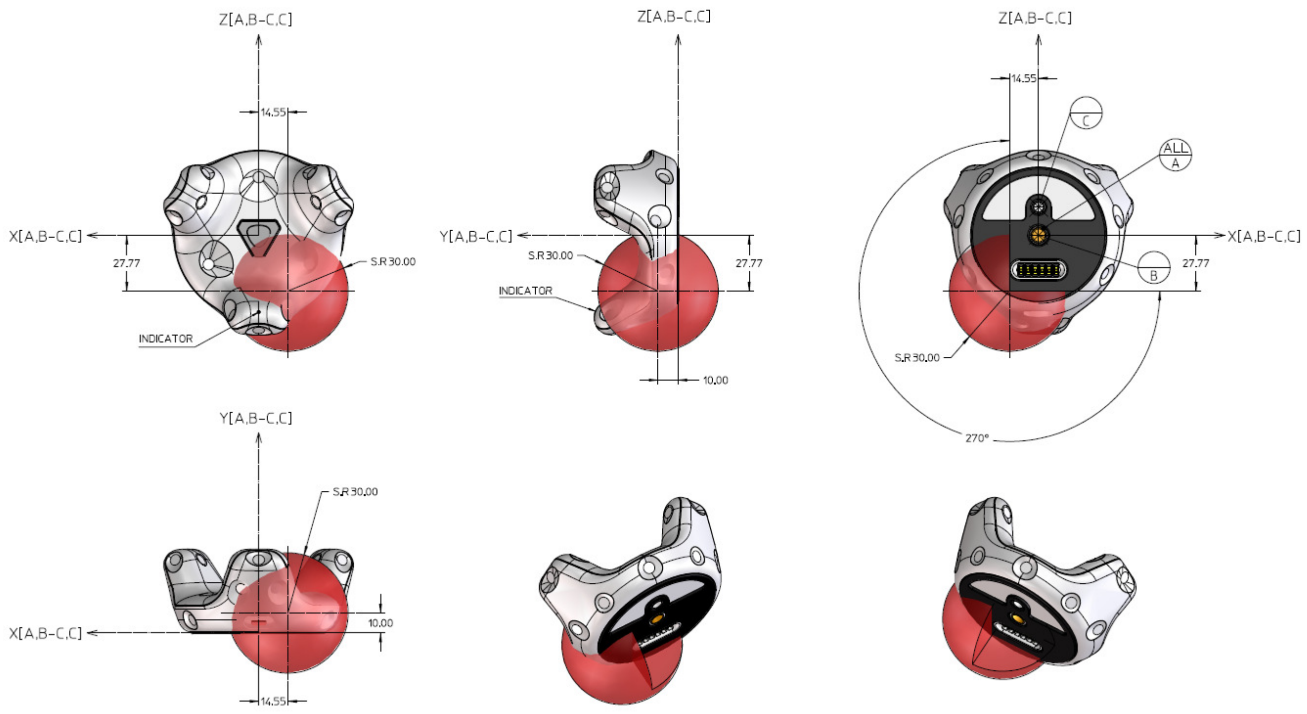


Figure: Restricted Area of Antenna

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

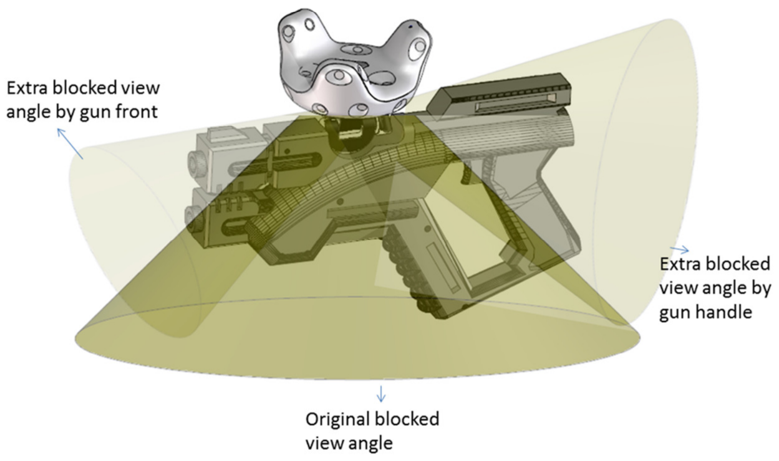

Extra blocked view angle by gun front
Extra blocked view angle by gun handle
Original blocked view angle

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 the tracker, and the other one holds the accessory.

c. The user should not be physically harmed while attaching or detaching the tracker.

d. The user should be comfortable while attaching and detaching the tracker.

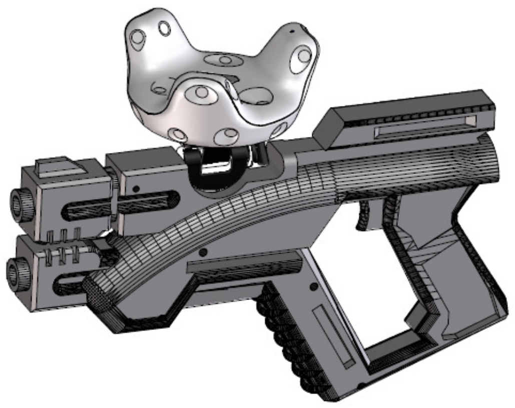
e. The form the accessory attached with the tracker should consist of the shape of the physical object to avoid hitting during operating.

f. The tracker should not be blocked by the accessory and affect the tracking performance.

g. The outer skin of the accessory must be low reflecting to avoid optical performance influence.

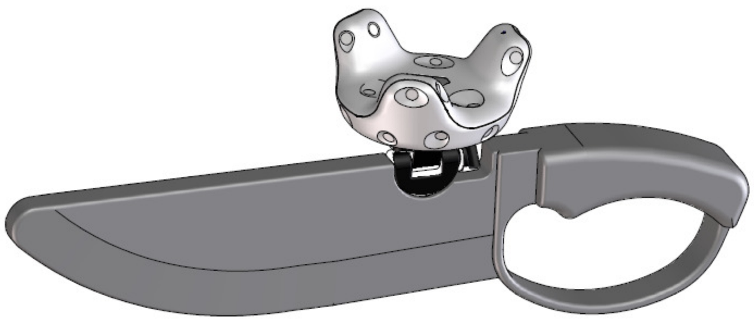
Docking embodiments

Gun



Sword

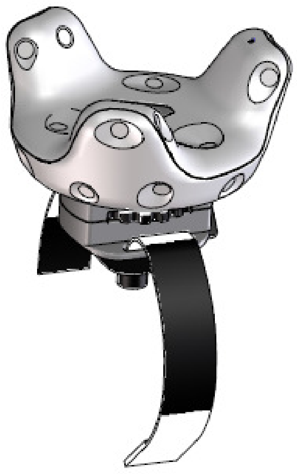
It is recommended to design the mounting mechanism close to the hand-held area, and set up the length in the VR program.



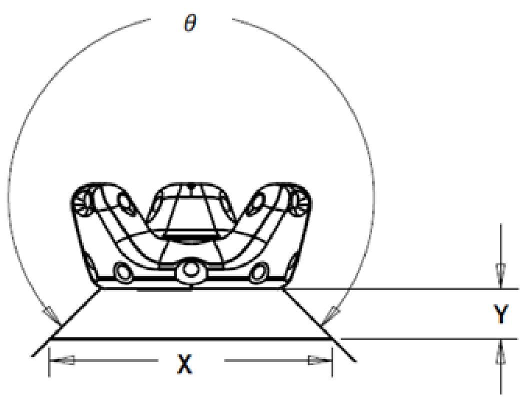
Multi-purpose docking base

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:

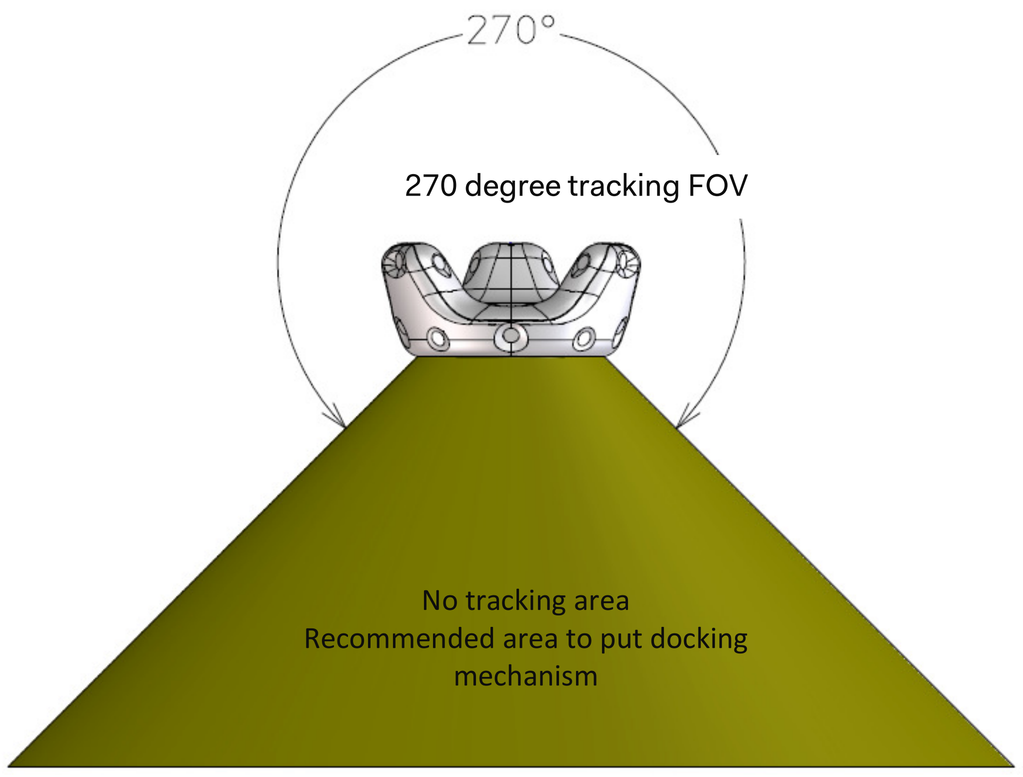


$$\theta = 180^\circ + 2 \times tan^{-1}\frac{2Y}{X}$$

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.

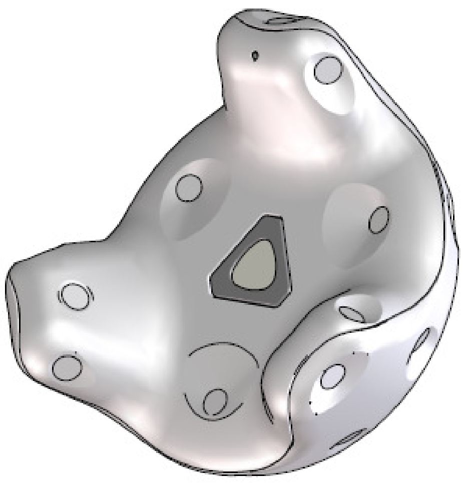


Figure: Vive Tracker

Apparel size

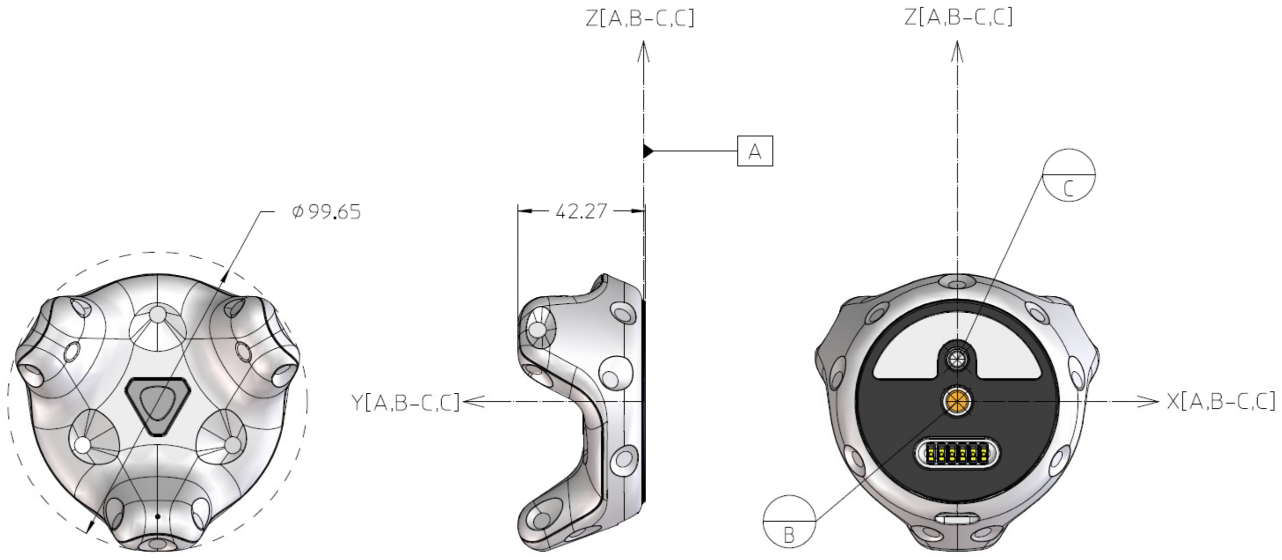


Figure: Vive Tracker with different angle

The overall size of the Vive Tracker is Φ99.65mm * 42.27mm (H).

Main feature

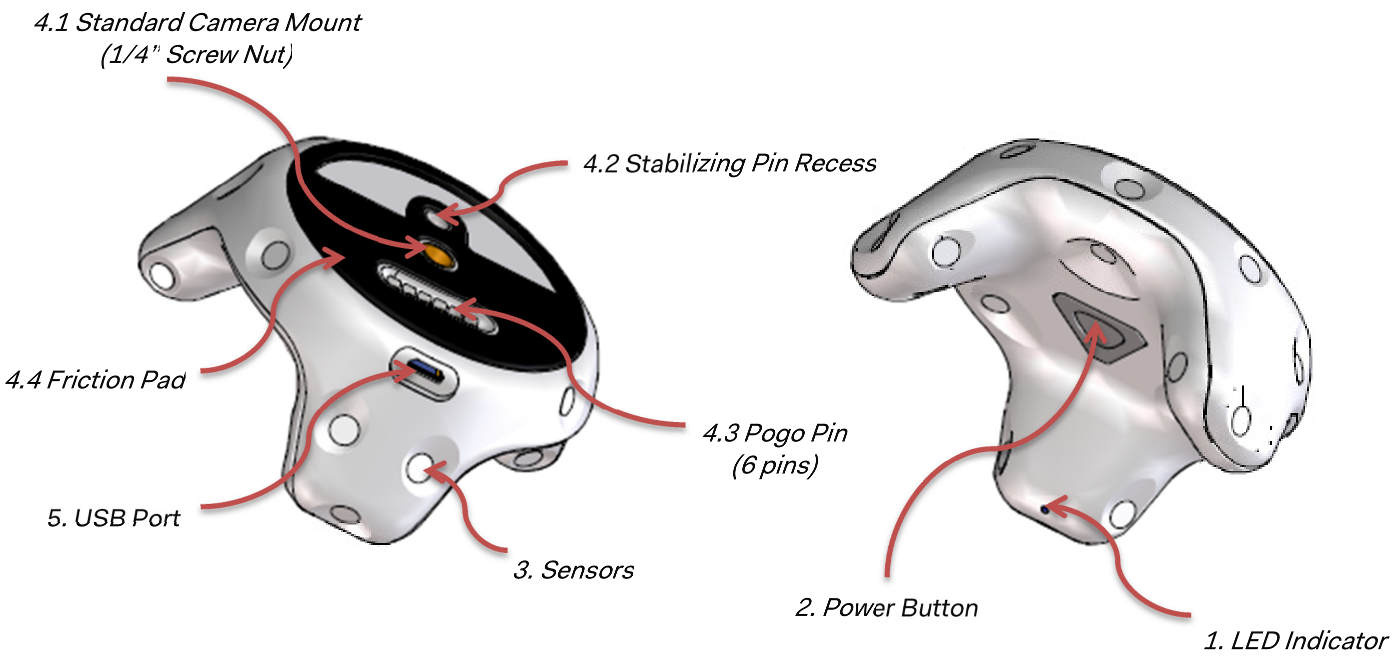


Figure: Main features

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 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)

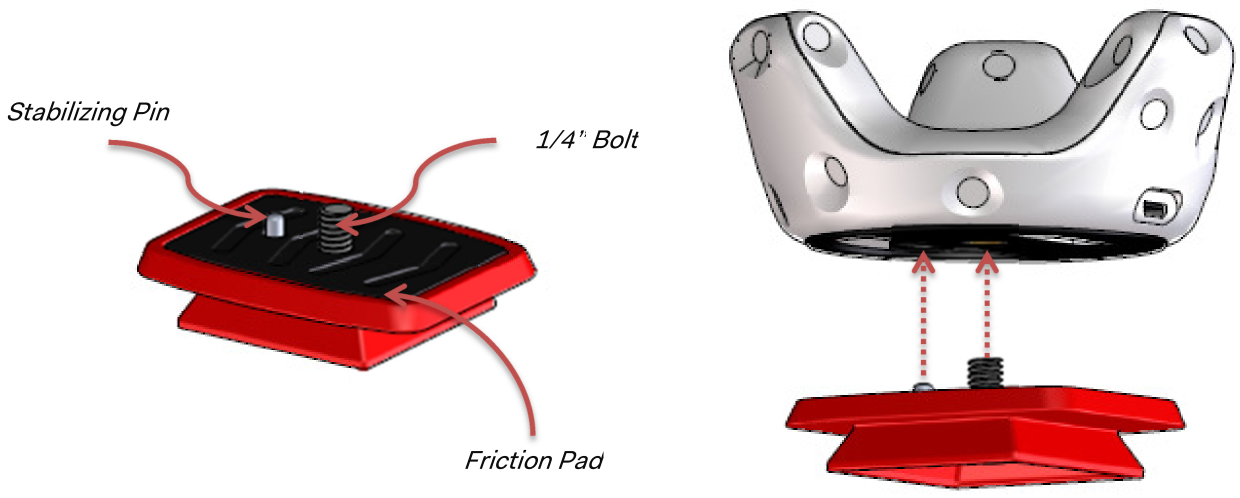


Figure: Docking with standard tripod cradle

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)

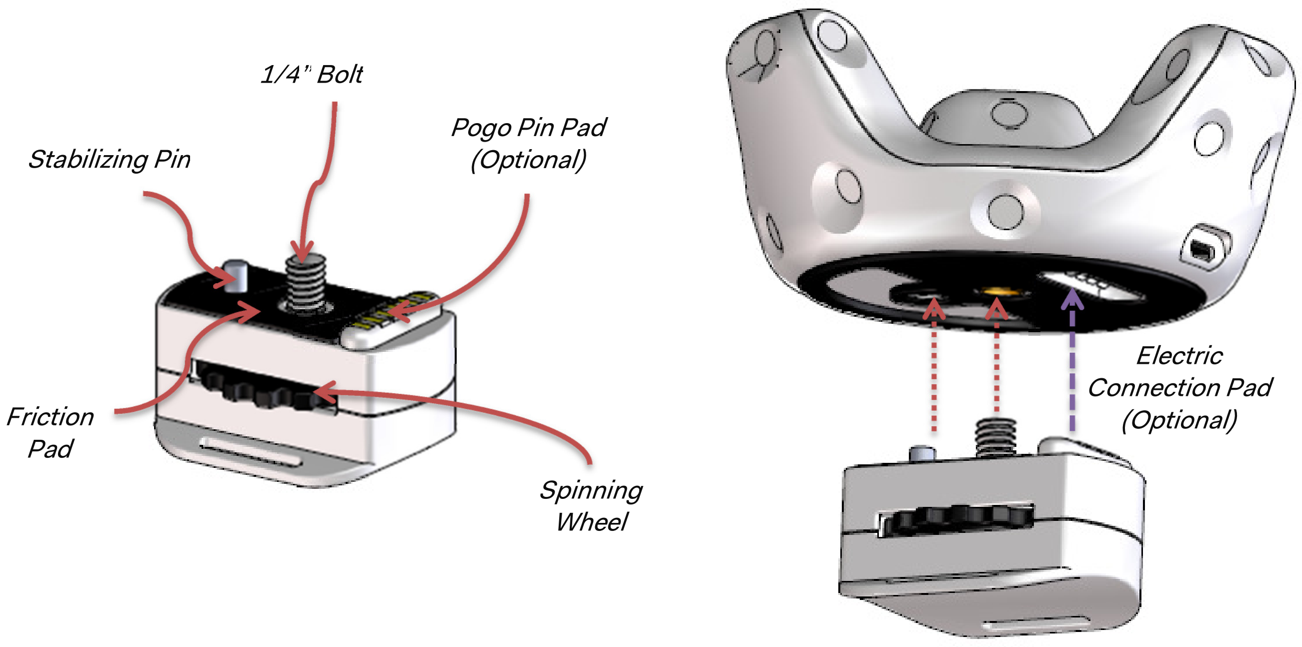


Figure: Docking with side tightening wheel

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:

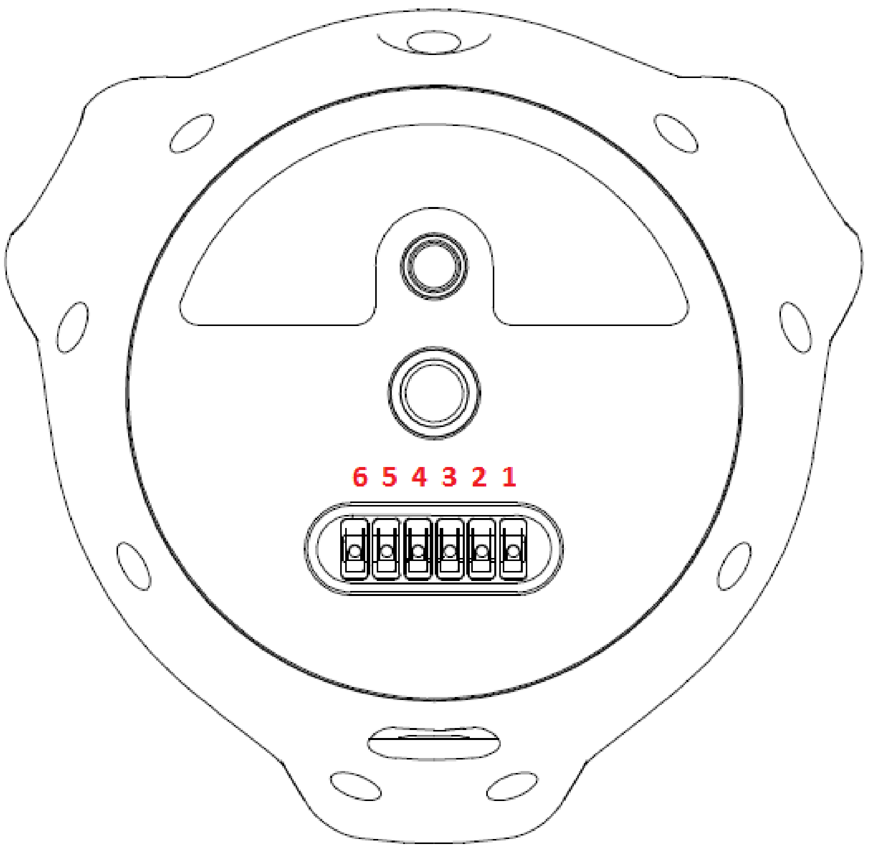
- 1/4” bolt design
  Please refer to ISO 1222-2010, Figure 1 on page 1.

- 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 .

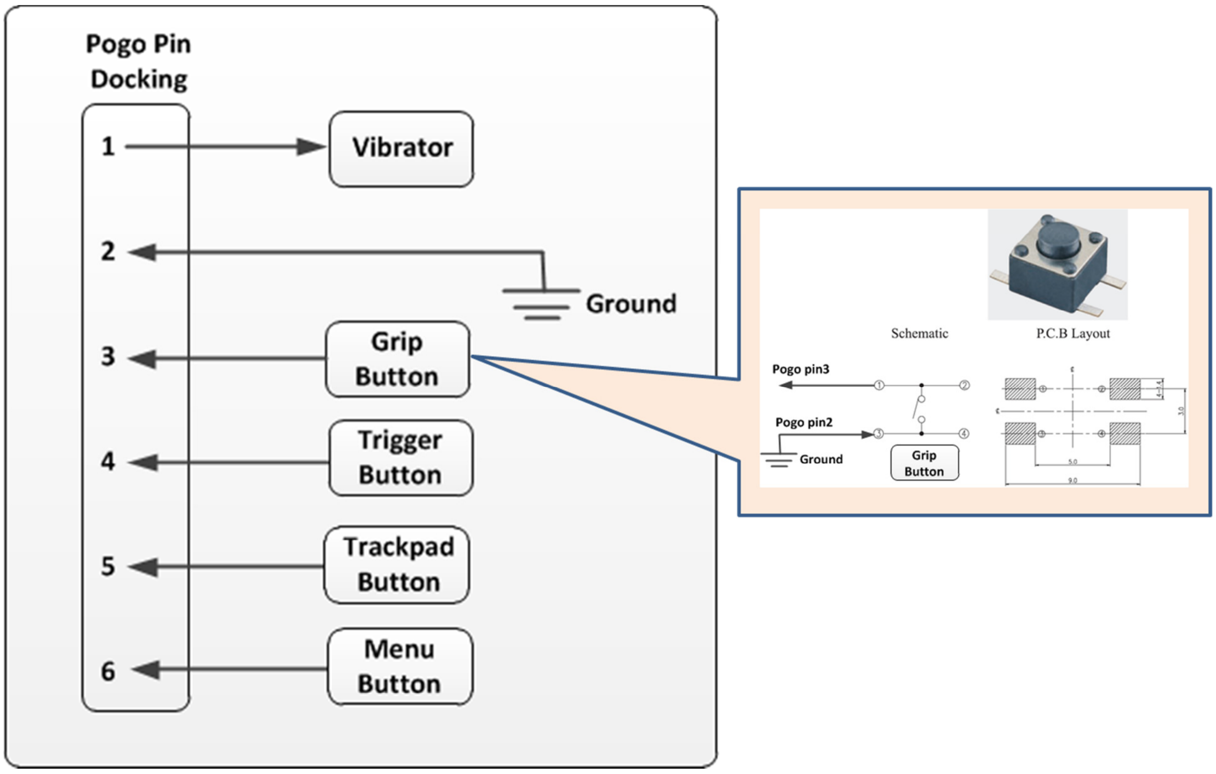
- Screw thread design
  The screw thread type that applies to Vive Tracker is 1/4” screw with 1.27 mm pitch. For detailed information, please refer to ISO 1222-2010, pages 3-5.

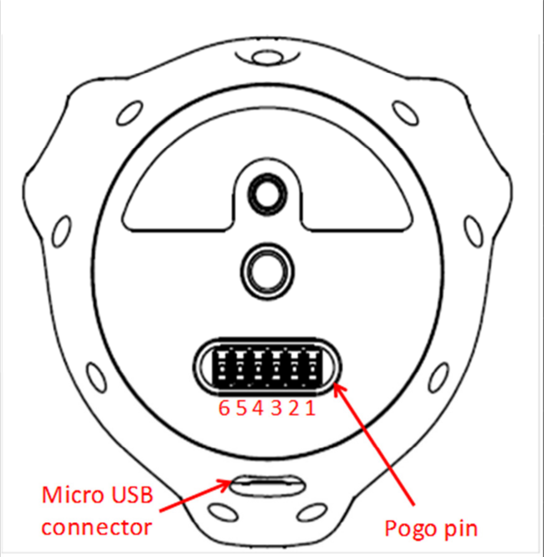
Design of Pogo Pin Pad

a. Pin definition (Vive Tracker)



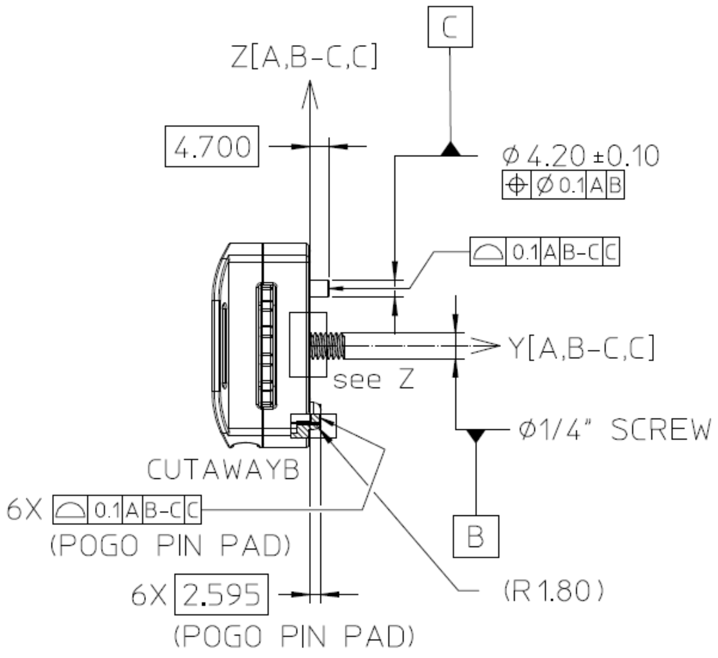
b. Pogo Pin Pad reference design
Electrical

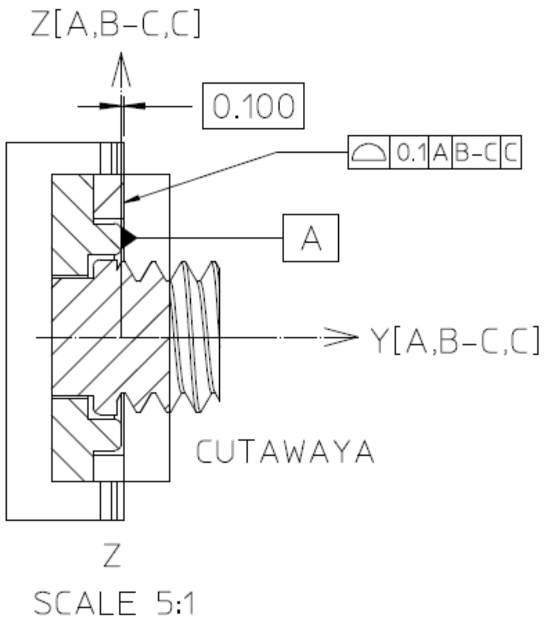


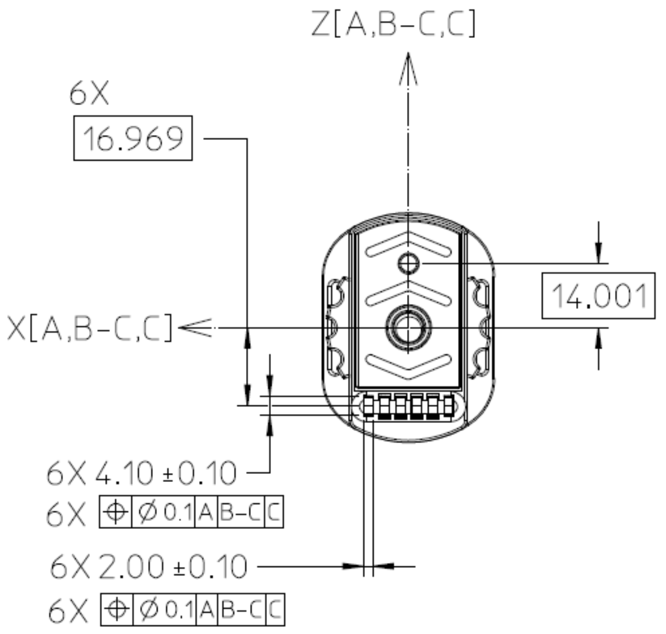


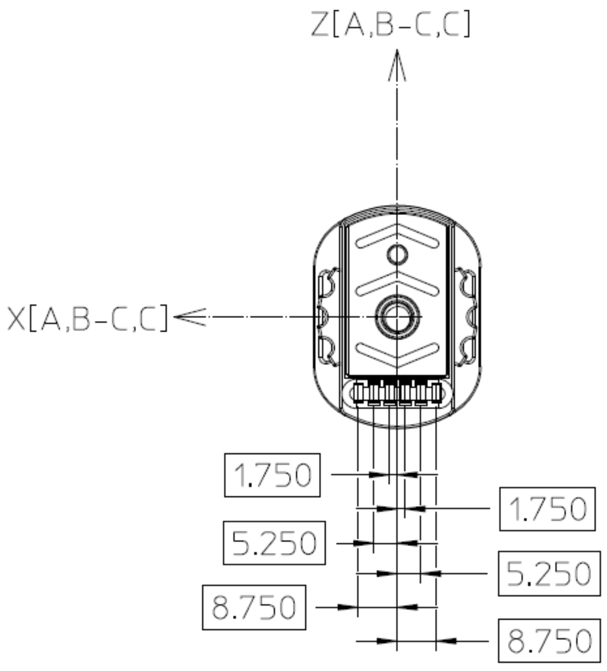
Pin no.	Type	Description
1	Digital output	General purpose output pin
2	GND	Ground
3	Digital / Power input	General purpose input pin: Internal pull up resistor to VDD, Active-low (Grip button) Power input pin
4	Digital input	General purpose input pin: Internal pull up resistor to VDD, Active-low (Trigger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

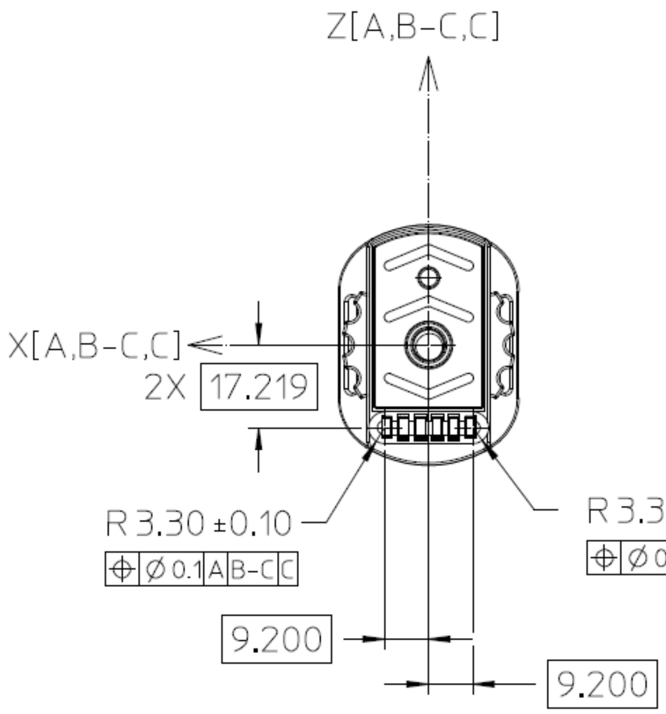
Mechanical

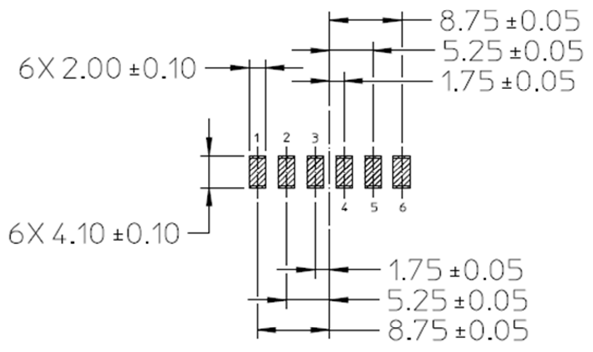

C
Z[A,B-C,C]
4.700
Ø4.20 ±0.10
⌖ Ø0.1 A B
⌓ 0.1 A B-C C
Y[A,B-C,C]
see Z
Ø1/4" SCREW
CUTAWAYB
6X ⌓ 0.1 A B-C C
(POGO PIN PAD)
B
6X 2.595
(POGO PIN PAD)
(R1.80)


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 :

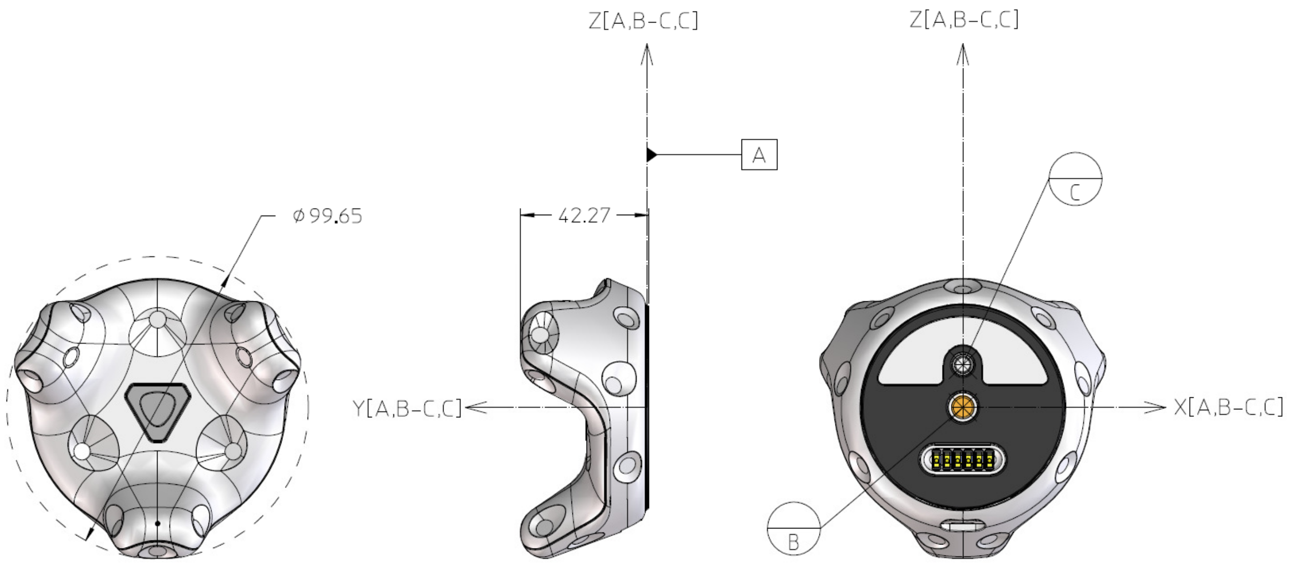
1.8A (Min)

Coordinate system

The coordinate system of Vive Tracker is "**Right-handed coordinate system**".

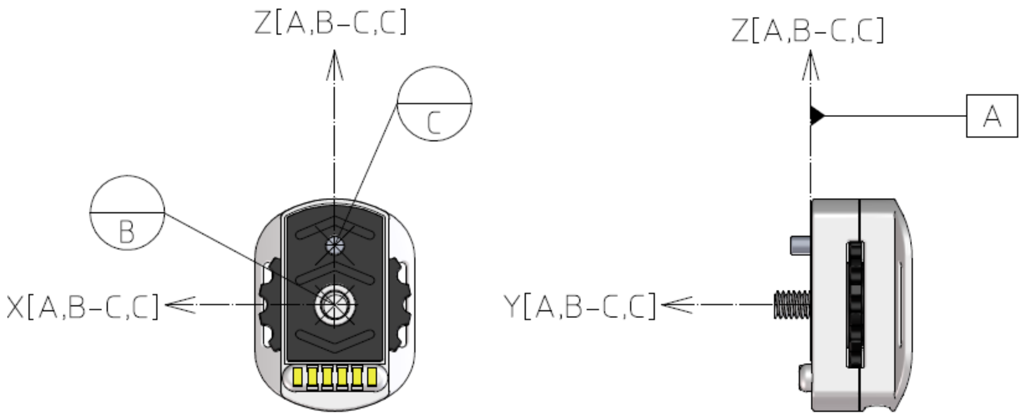
Vive Tracker

- 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

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.

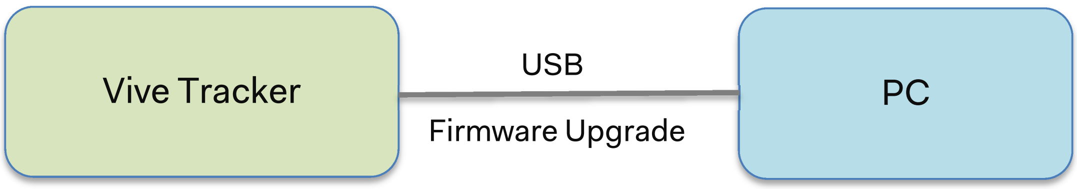


Figure: Vive Tracker and PC

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
   Please 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.

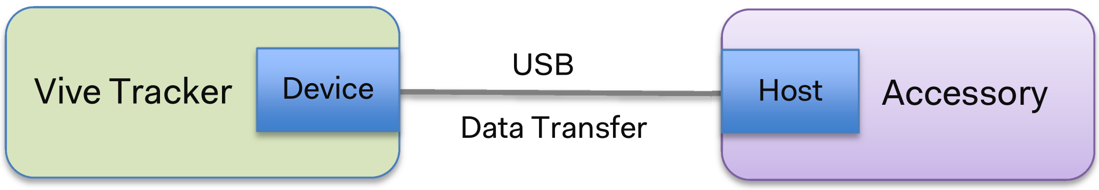


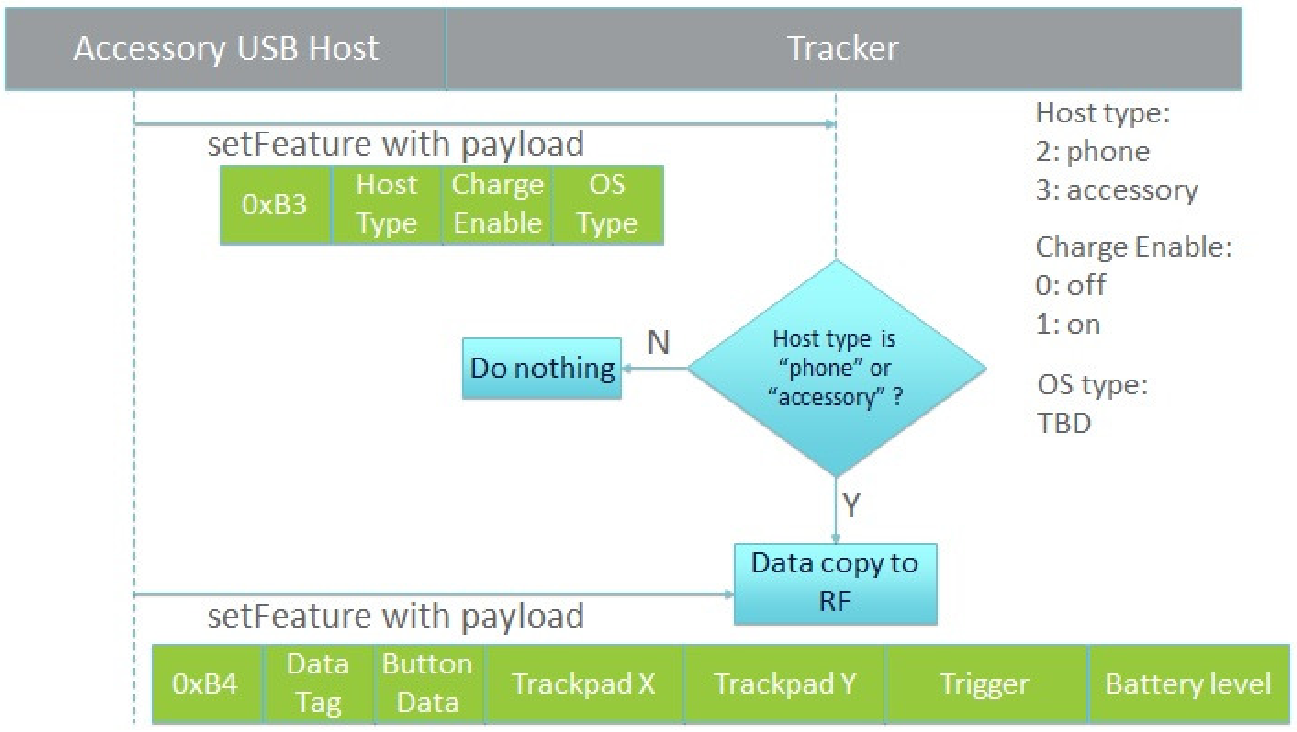
Figure: Vive Tracker and accessory

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.

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
3		
4	Pad Y value	Pad Y value, value from -32768 to 32767
5		
6	Trigger Raw	Trigger Raw, value from 0 to 65535
7		
8	Battery Level	Battery Level, Reserved
9		

Table: Data Format (Accessory to Vive Tracker)

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 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:

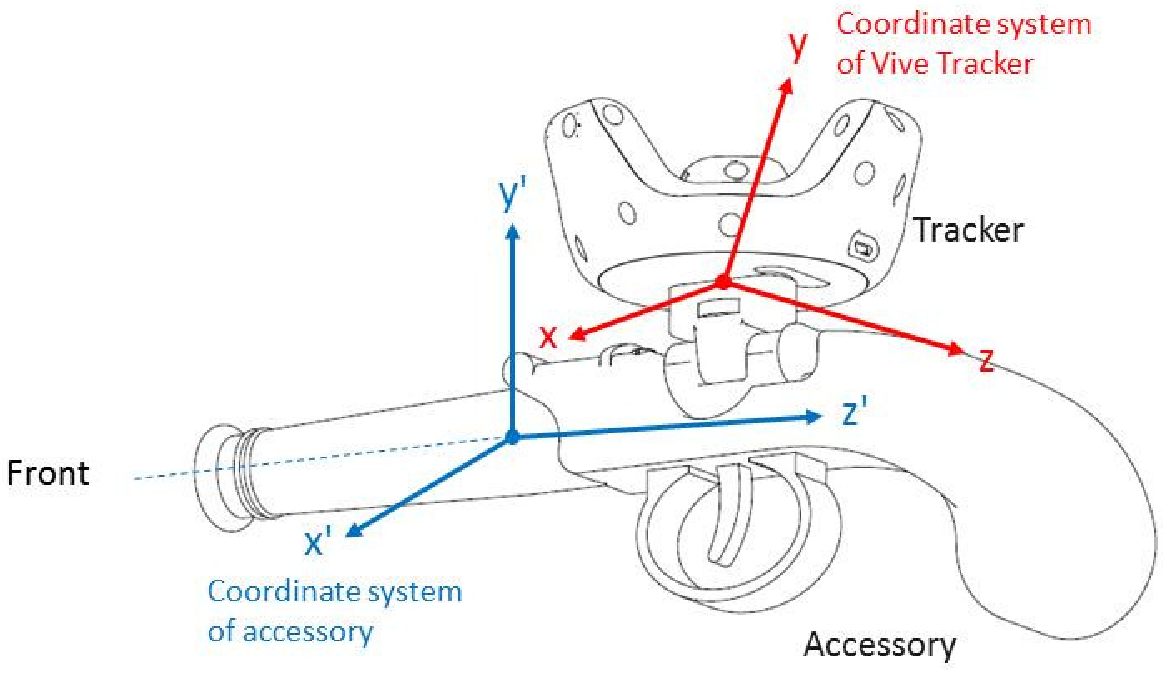

y
Coordinate system of Vive Tracker
y'
Tracker
x
z
z'
Front
x'
Coordinate system of accessory
Accessory

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 example 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 example code for accessory integration (1)

Another Unity example 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 example 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 add Vive Tracker into SteamVR. Assume that you have two Vive controllers already, and you have plugged in the dongle into the dongle cradle to PC'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 paring mode.

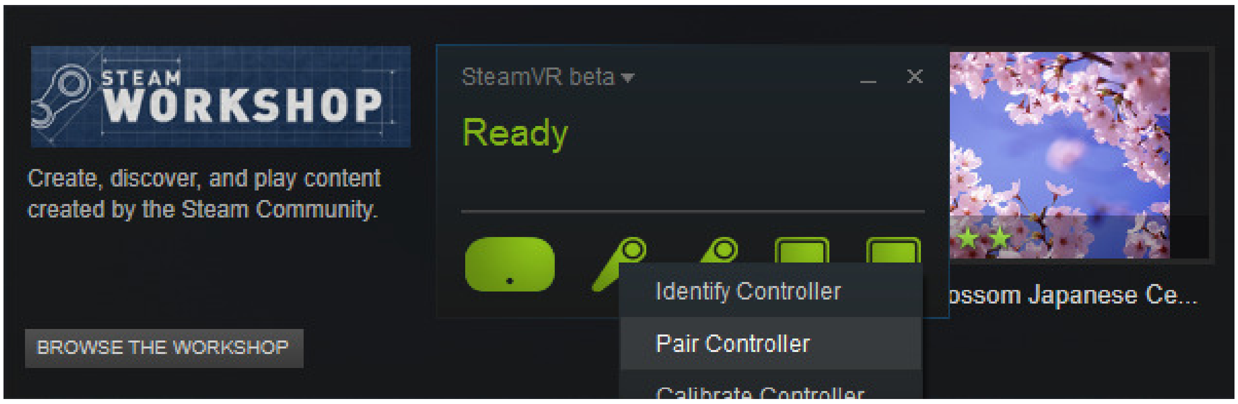


Figure: Pair Vive Tracker

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

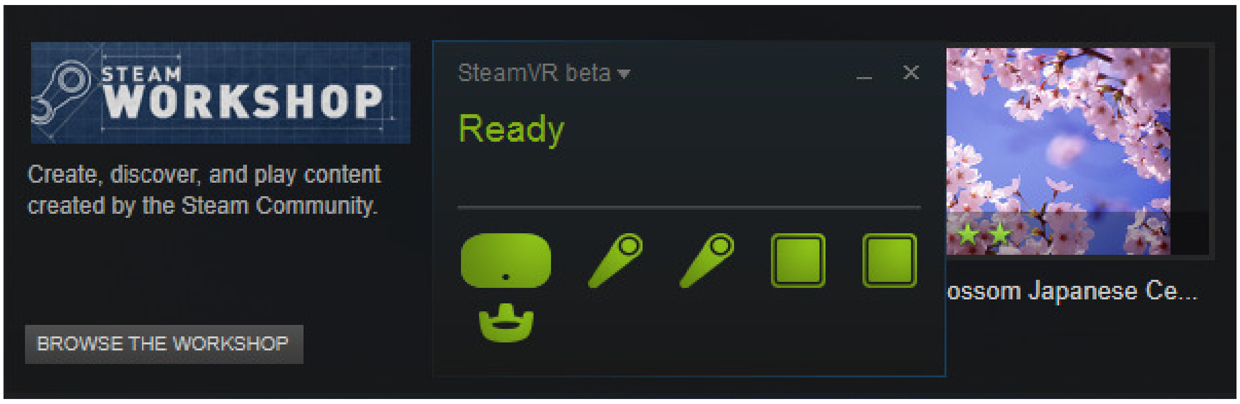
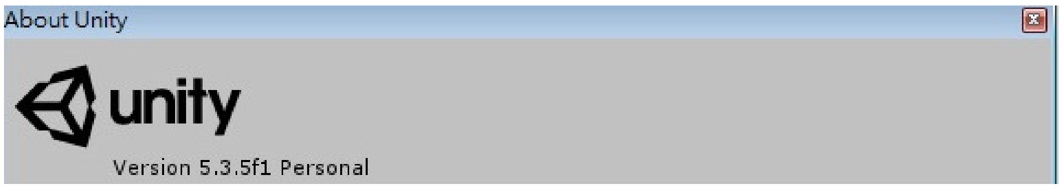


Figure: Vive Tracker is added in SteamVR

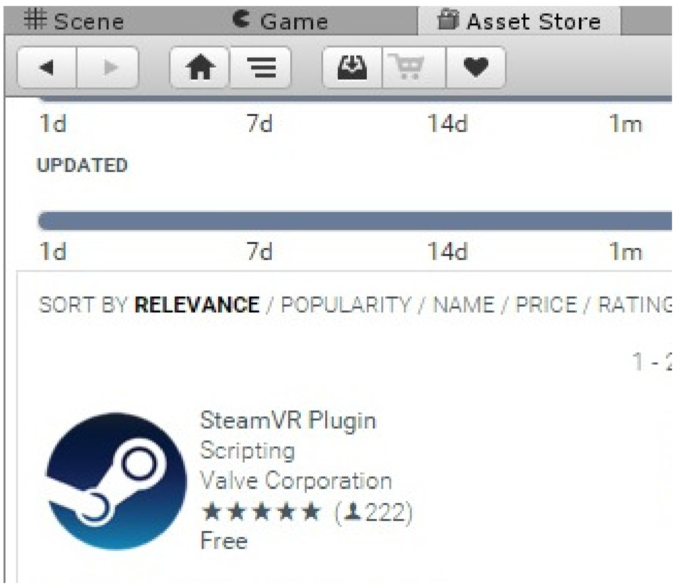
You can download Unity is 5.3.5f1 Personal from
https://store.unity.com/download?ref=personal



Version 5.3.5f1 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.

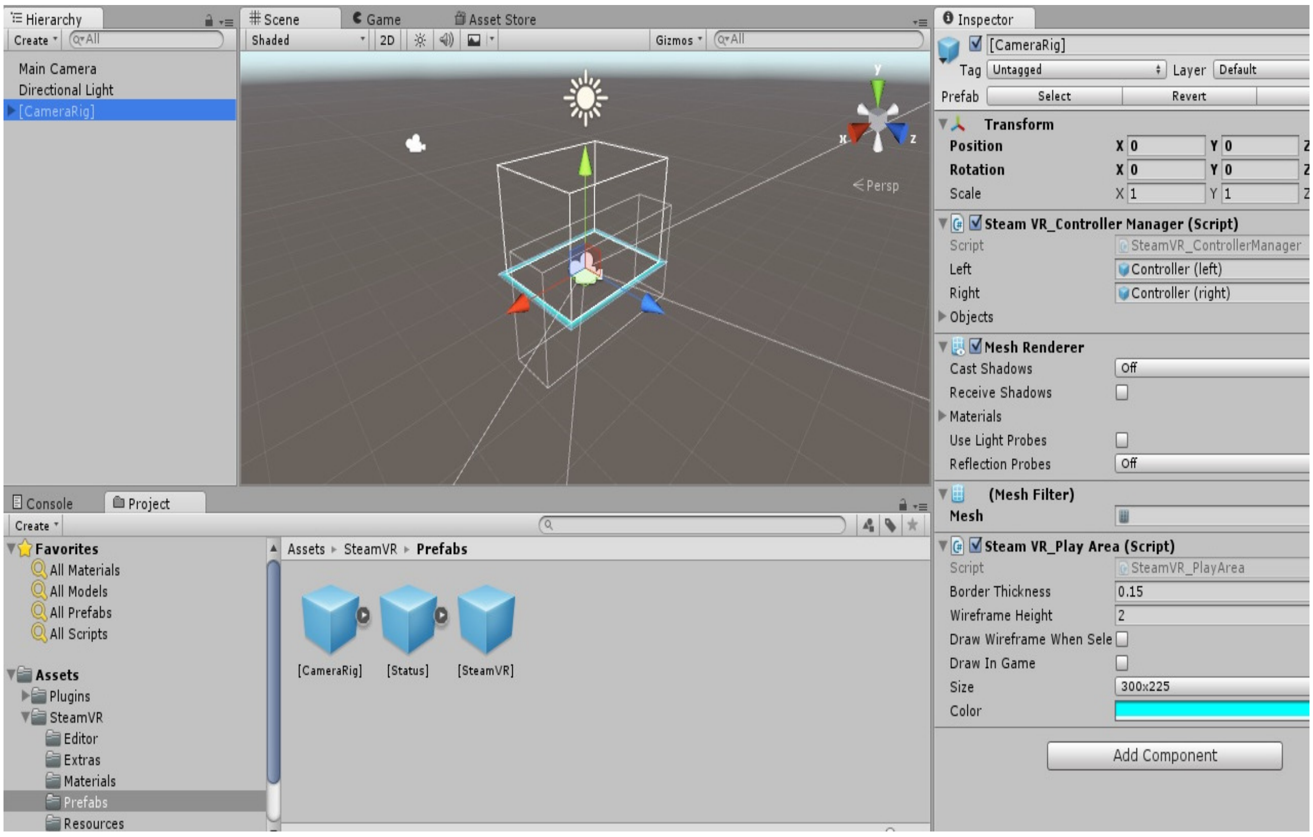


SteamVR Plugin
Scripting
Valve Corporation
★★★★★ (222)
Free

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.



[CameraRig]
Tag Untagged Layer Default
Prefab
Transform
Position X 0 Y 0
Rotation X 0 Y 0
Scale X 1 Y 1
Steam VR_Controller Manager (Script)
Script SteamVR_ControllerManager
Left Controller (left)
Right Controller (right)
Objects
Mesh Renderer
Cast Shadows Off
Receive Shadows
Materials
Use Light Probes
Reflection Probes Off
(Mesh Filter)
Mesh
Steam VR_Play Area (Script)
Script SteamVR_PlayArea
Border Thickness 0.15
Wireframe Height 2
Draw Wireframe When Sele
Draw In Game
Size 300x225
Color

Figure: Add "CameraRig"

Step 2: Create 3D Object for Vive Tracker. In this example, "Capsule" is used.

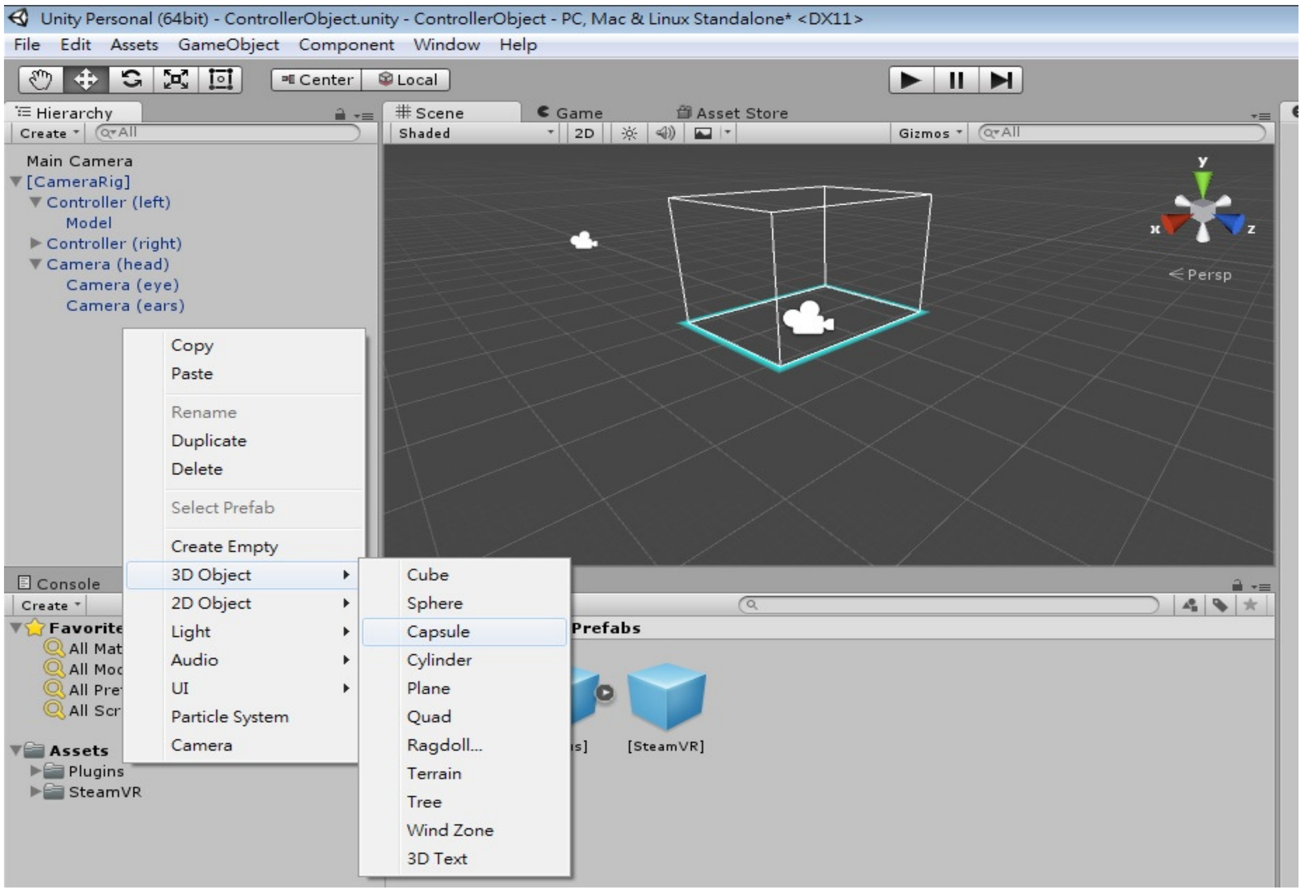


Figure: Create 3D Object

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

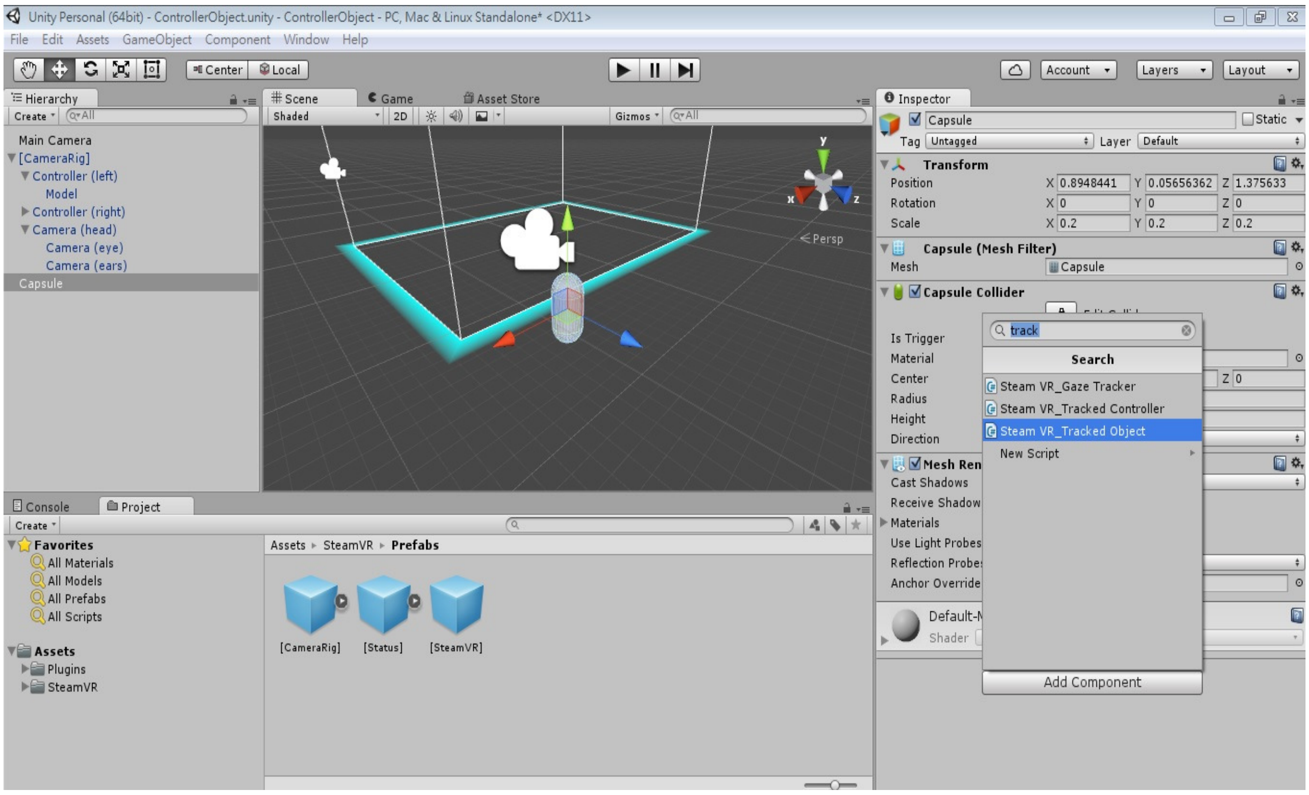
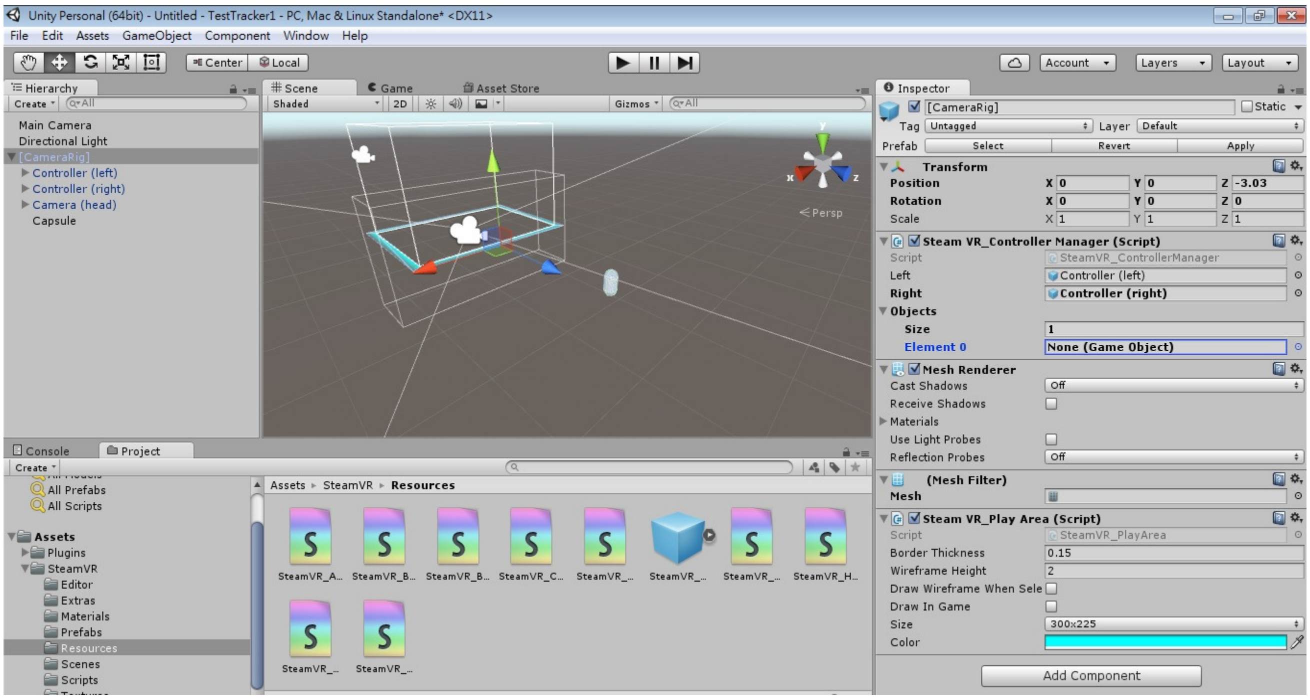


Figure: Add Component “SteamVR_Tracked Object”

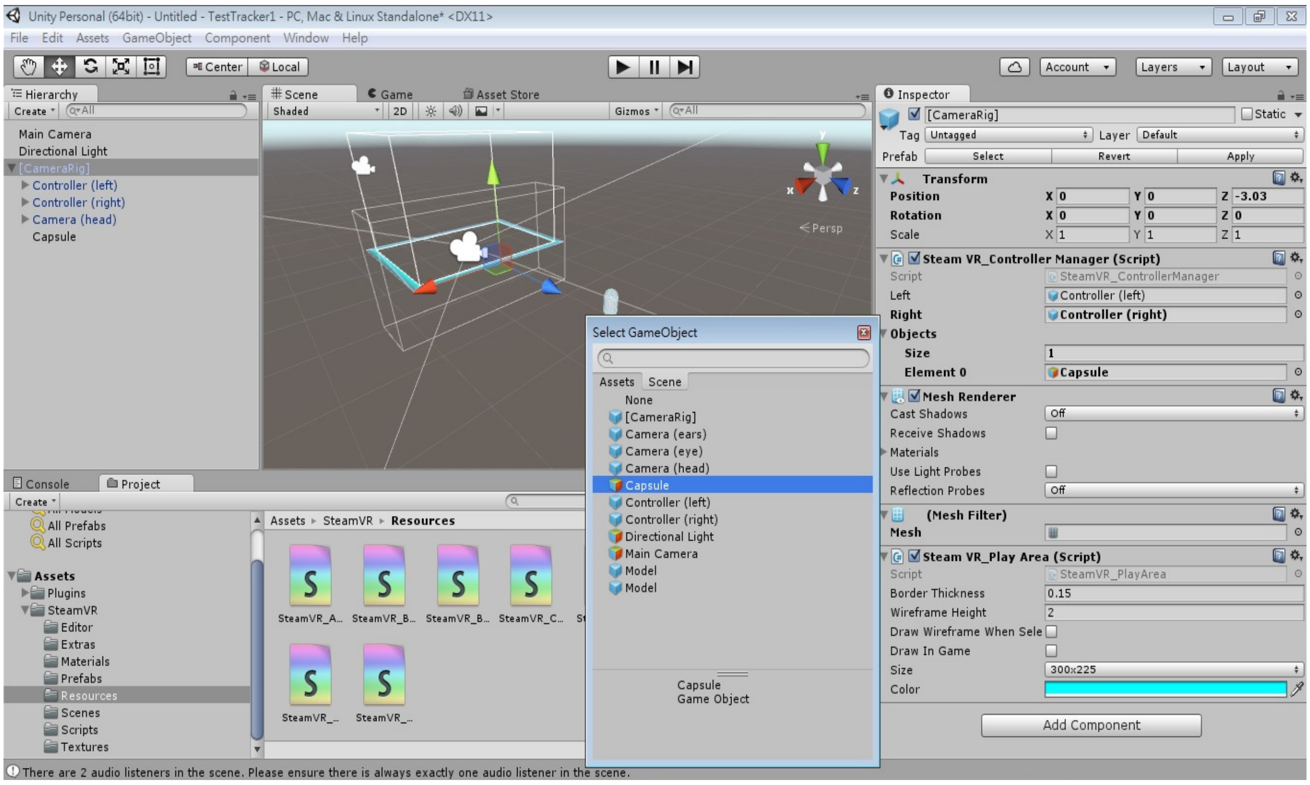
Step 4: Set size of Objects item in “SteamVR Controller Manager”. In this example, one Vive Tracker is used in the setup.



```
[CameraRig]                                Static
Tag Untagged          Layer Default
Prefab   Select   Revert   Apply
Transform
Position     X 0    Y 0    Z -3.03
Rotation     X 0    Y 0    Z 0
Scale        X 1    Y 1    Z 1
Steam VR_Controller Manager (Script)
Script       SteamVR_ControllerManager
Left         Controller (left)
Right        Controller (right)
Objects
  Size       1
  Element 0  None (Game Object)
Mesh Renderer
Cast Shadows          Off
Receive Shadows
Materials
Use Light Probes
Reflection Probes     Off
(Mesh Filter)
Mesh
Steam VR_Play Area (Script)
Script                   SteamVR_PlayArea
Border Thickness         0.15
Wireframe Height         2
Draw Wireframe When Sele
Draw In Game
Size                     300x225
Color
```

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

Step 5: Set “Capsule” object to “Element0” on the “Objects” item in “SteamVR Controller Manager”.



[CameraRig] Static
Tag Untagged Layer Default
Prefab Select Revert Apply

Transform
Position X 0 Y 0 Z -3.03
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Steam VR_Controller Manager (Script)
Script SteamVR_ControllerManager
Left Controller (left)
Right Controller (right)
Objects
Size 1
Element 0 Capsule

Mesh Renderer
Cast Shadows Off
Receive Shadows
Materials
Use Light Probes
Reflection Probes Off

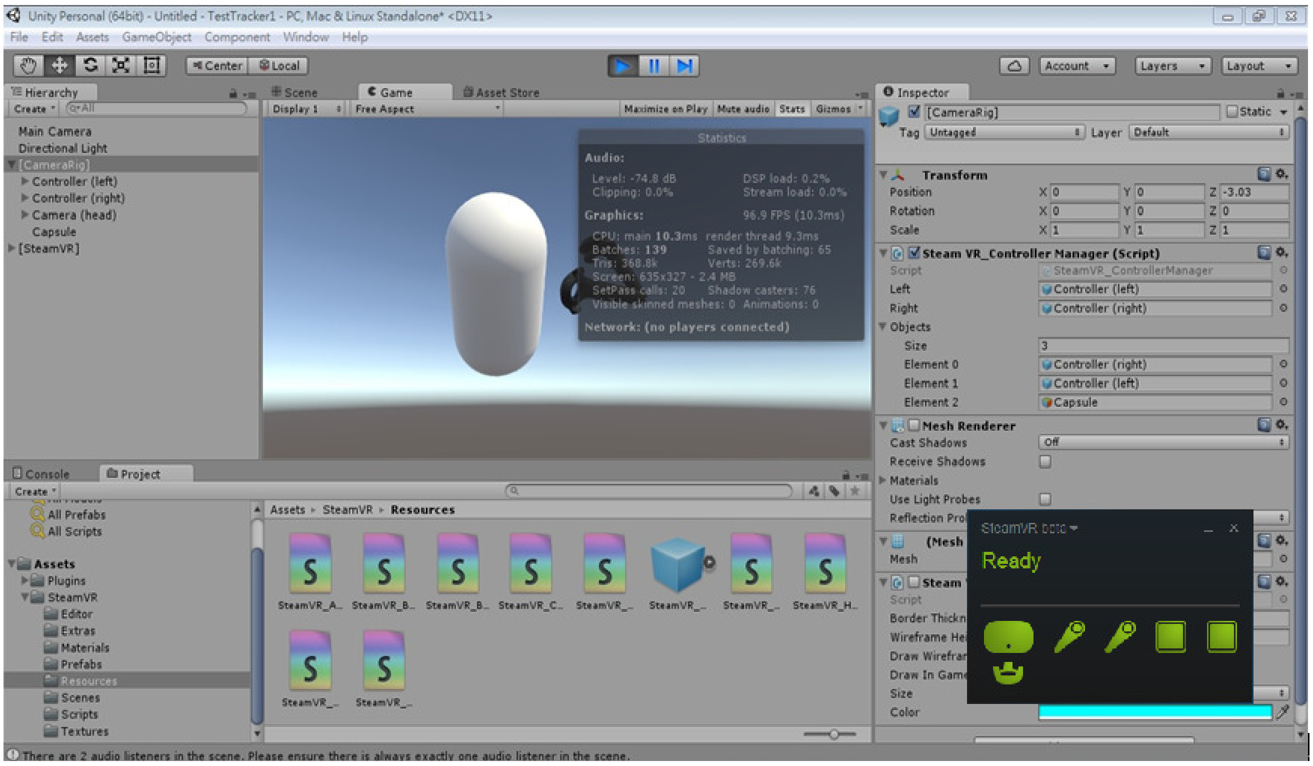
(Mesh Filter)
Mesh

Steam VR_Play Area (Script)
Script SteamVR_PlayArea
Border Thickness 0.15
Wireframe Height 2
Draw Wireframe When Sele
Draw In Game
Size 300x225
Color

There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.

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.



```
Statistics
Audio:
  Level: -74.8 dB          DSP load: 0.2%
  Clipping: 0.0%           Stream load: 0.0%
Graphics:                  96.9 FPS (10.3ms)
  CPU: main 10.3ms  render thread 9.3ms
  Batches: 139        Saved by batching: 65
  Tris: 368.8k        Verts: 269.6k
  Screen: 635x327 - 2.4 MB
  SetPass calls: 20   Shadow casters: 76
  Visible skinned meshes: 0  Animations: 0
Network: (no players connected)
```

```
[CameraRig]                       Static
Tag Untagged   Layer Default
Transform
  Position   X 0   Y 0   Z -3.03
  Rotation   X 0   Y 0   Z 0
  Scale      X 1   Y 1   Z 1
Steam VR_Controller Manager (Script)
  Script     SteamVR_ControllerManager
  Left       Controller (left)
  Right      Controller (right)
  Objects
    Size       3
    Element 0  Controller (right)
    Element 1  Controller (left)
    Element 2  Capsule
Mesh Renderer
  Cast Shadows      Off
  Receive Shadows
  Materials
  Use Light Probes
  Reflection Pro
(Mesh
  Mesh
Steam
  Script
  Border Thickn
  Wireframe Hei
  Draw Wirefran
  Draw In Game
  Size
  Color
```

```
There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.
```

Figure: Execution of Unity

Firmware upgrade

In the developer version 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 the device in DVT stage, please follow below steps to upgrade the firmware with upgradeFirmware.zip file first.

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

1. Unplug Vive controller if it connect to PC via USB.
2. Connect one Vive Tracker with that PC by using a USB cable.
3. Unzip the file we attached upgradeFirmware.zip
4. Execute the file upgradeFirmware.exe
5. When it done, remove cable to reboot device
6. You will see Vive Tracker icon with SteamVR beta

For the device after DVT stage, follow SteamVR notification to update or follow below steps to upgrade the firmware of Vive Tracker:

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. Unplug Vive controller if it connect to PC via USB.
3. Connect one Vive Tracker with that PC by using a USB cable.
4. Execute the commands in a command window to update firmware.

   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
   ```