

Session 5

Input and Hand Using Controllers

Welcome to the session, Input and Hand Using Controllers.

This session gives information on how to use hand controller in VR.

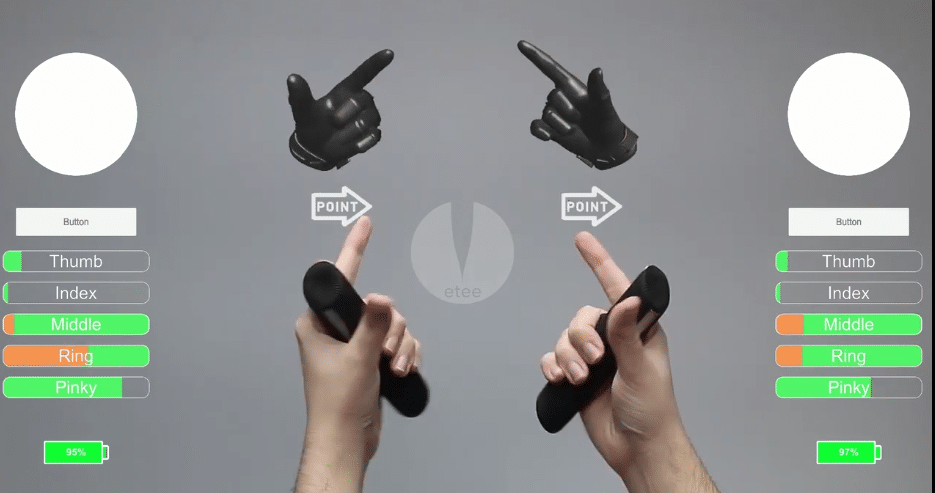
In this session, students will learn to:

* Describe hand controllers
* Explain hands prefab in OculusSDK
* Explain how to create controller as HandController

5.1 Using Hands as VR Controller

Using hands as **virtual reality (VR) controllers** in Unity can provide a more immersive experience for the user by allowing them to interact with virtual objects using their own hands. To implement this, you can use hand-tracking technology.

Using hands as controllers in VR can be very helpful for several reasons as shown in Figure 5.1:



**Figure 5.1 VR Controller as Hands**

1. **Immersion**:By using your hands to interact with objects in a virtual environment, you can create a more **immersive experience**.You can reach out and touch objects as if they were really there, making the experience more realistic and engaging.
2. **Intuitive controls**: Hand tracking can make controls more intuitive and natural. Instead of having to use a controller to simulate hand movements, you can use your **hands.**
3. **Fine motor control**: Hands are incredibly dexterous, which makes them well-suited for performing fine **motor tasks** in VR. This can include things such as manipulating small objects, adjusting controls, or typing on a virtual keyboard.
4. **Accessibility**: Using hands as controllers can also make VR more accessible to people with physical disabilities. For example, someone who is unable to grip a controller with their hands may still be able to use hand tracking to interact with objects in a virtual environment.
5. **Gesture-based interactions**:Hands offer a rich vocabulary of gestures that can be used for **intuitive interactions** in VR. For example, you can point at objects to select them, grab and move objects with natural hand motions, or make gestures to activate specific features or actions.
6. **Faster interaction**: Using hands as controllers can also allow for faster and more fluid interactions. With **traditional controllers,** there can be a delay between the player’s actions and the on-screen response. With hand tracking, the response can be immediate, allowing for more rapid and natural interactions.
7. **Reduced cognitive load**: When using traditional controllers in VR, players may have to constantly think about which button or joystick to use for a specific action. With hand tracking, the controls are more **intuitive, reducing the cognitive** load and allowing for a more immersive experience.
8. **More natural feedback**: By using hands as controllers, players can receive a more natural feedback when interacting with objects in VR. For example, when you pick up a **virtual object**, you can feel the weight and resistance of the object in your hand, which can enhance the sense of presence and immersion.
9. **Cost savings**:Using hands as controllers can also save on hardware costs. Traditional VR controllers can be expensive to manufacture, and they require batteries or other power sources. Hand tracking, on the other hand, can be done using the existing sensors on many **VR headsets**, which can save on hardware costs and reduce the requirement for additional batteries.

5.2 What are Hands Prefab in Oculus Interaction SDK?

In Unity VR with **Oculus Integration SDK**, the hands prefab is a pre-made game object that includes a hand model and a script for tracking hand movements and gestures in the **VR environment**. The hands prefab is provided by the Oculus Integration SDK as a quick and easy way to add hand tracking and input functionality to your VR project.

The hands prefab can be found in the **Oculus/SampleFramework/Hands** folder in the project panel. There are three prefabs available:

1. **OVRHandPrefab**: This prefab contains a realistic hand model that can be used to represent the player’s hands in the VR environment. It includes a mesh renderer, **an OVRHand** script, and a set of child objects representing the individual finger joints.
2. **OVRCustomSkeletonPrefab**: This prefab contains a **custom skeleton hand model** that can be used to create unique hand designs. It includes a mesh renderer, an OVRHand script, and a set of child objects that can be moved and rotated to create a custom hand shape.
3. **OVRCustomHandPrefab**: This prefab contains a **simplified hand model** that can be used to create custom hand shapes using primitives such as cubes and spheres. It includes a mesh renderer, an OVRHand script, and a set of child objects that can be moved and scaled to create a custom hand shape.

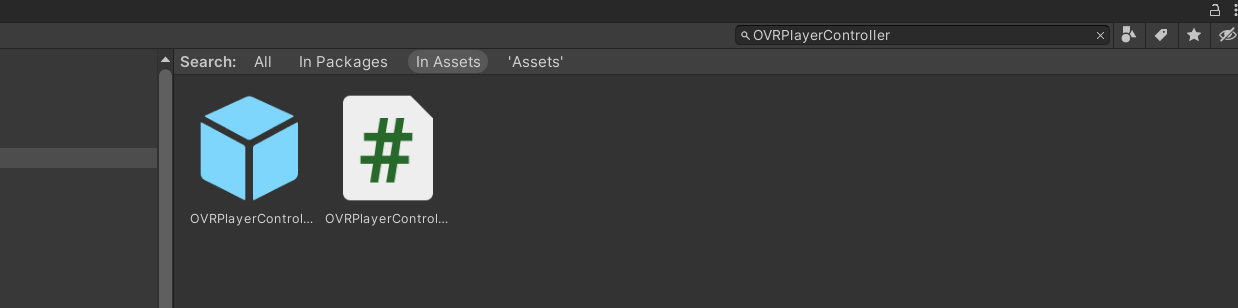
The hands prefab can be used to track the **position and rotation of the player**’**s hands** in the VR environment and to detect hand gestures such as grabbing and releasing objects. The OVRHand script attached to the hands prefab provides a set of public functions and events that can be used to customize the hand tracking and input functionality to fit your specific requirements.

5.3 Convert Hands as Controllers

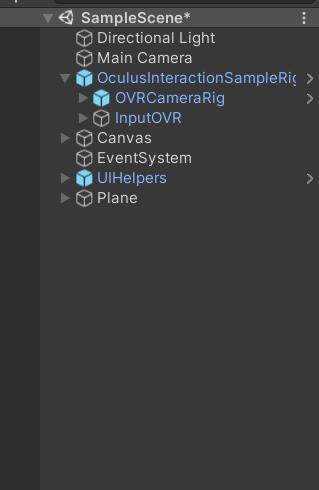
Converting hands into controllers in Unity involves using hand-tracking technology to track the movements and gestures of the user’s hands, and then mapping those movements and gestures to virtual controller inputs that can be used to interact with virtual objects in a Unity scene.

The steps to convert hands as controller are:

**Step 1**: Search and add **OVRPlayerController** in the scene as shown in Figure 5.2.

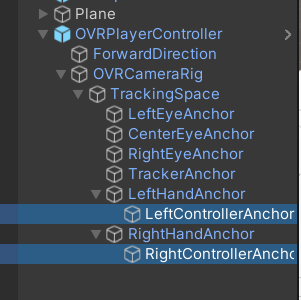
**Fig 5.2: Search OVRPlayerController**

**Step 2**: Add the **OVRCameraRig** prefab to your scene. This prefab provides a set of cameras that can be used to render the VR environment as shown in Figure 5.3.



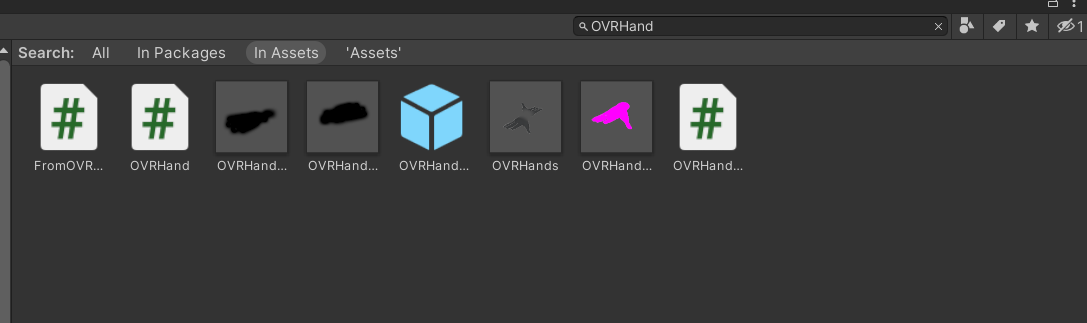
**Figure 5.3: Adding OVRCameraRig in Scene**

**Step 3**: Select the left and right touch controller objects in the **OVRPlayerController** prefab as shown in Figure 5.4.



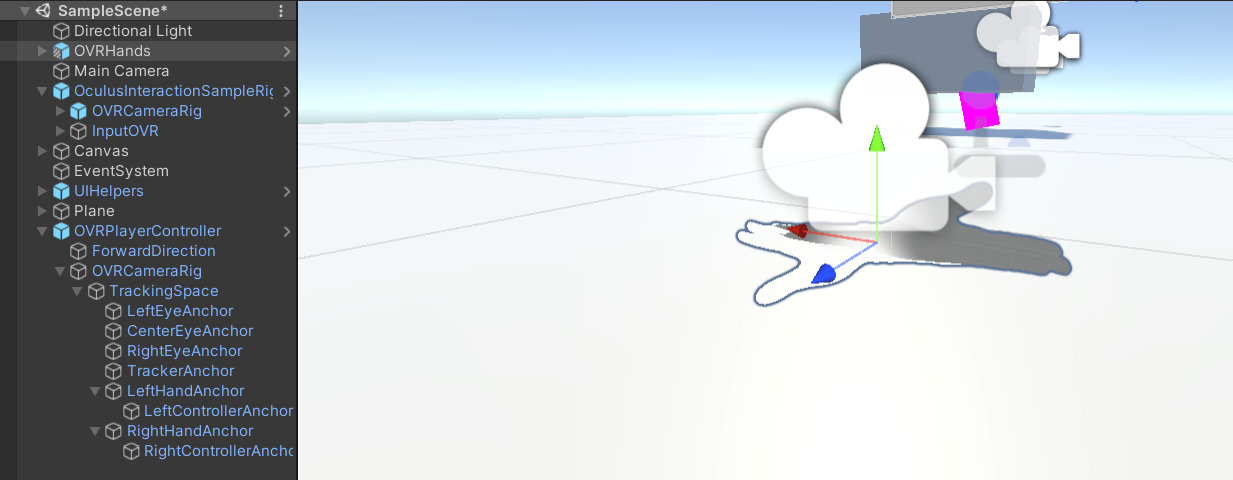
**Figure 5.4: OVRPlayercontroller → Left and RightControllerAnchor**

**Step 4**:Search for **OVRHands** in search panel as shown in Figure 5.5.



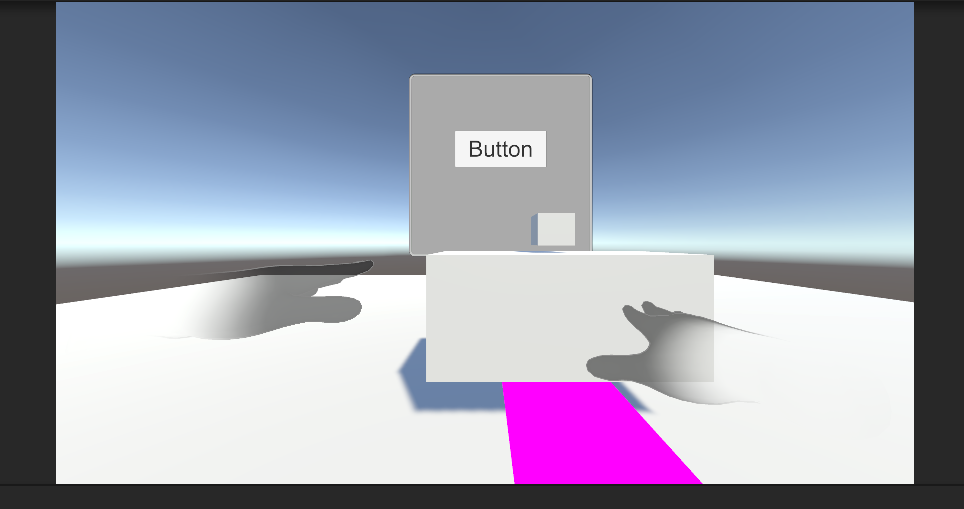
**Figure 5.5: OVRHands Search in Search Panel**

**Step 5**: Drag the **OVRHand** component to each of the controller objects. This component provides **hand tracking functionality** to the controller objects.Figure 5.6 shows OVRHand Controller added in scene.



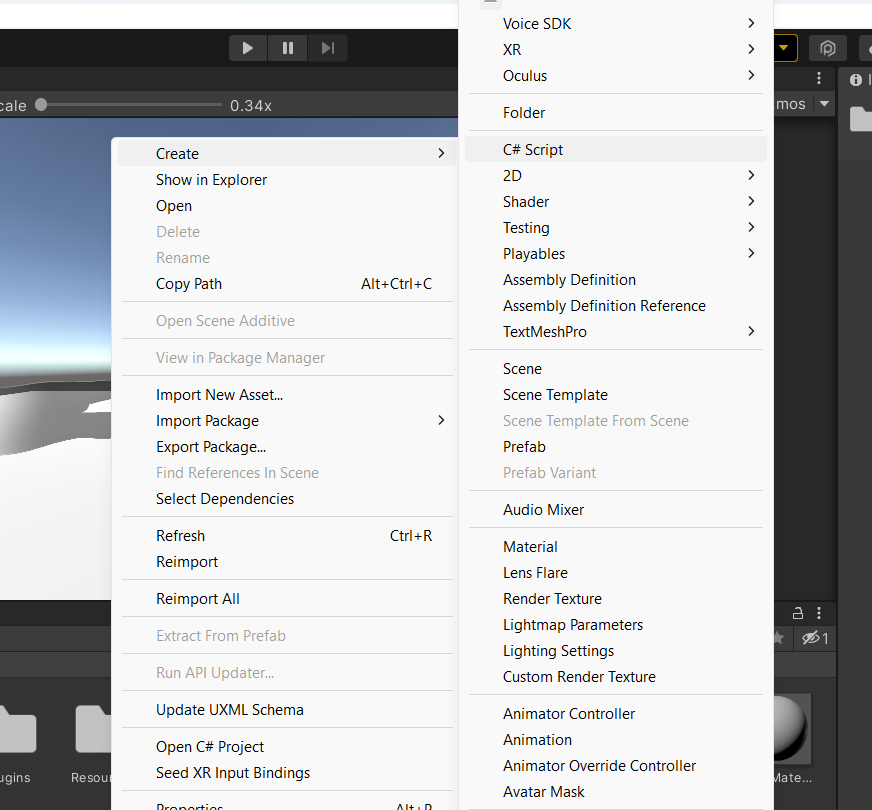
**Figure 5.6: OVRHand Controller Added in Scene**

**Step 6**:Set the hand property of each **OVRHand** component to the appropriate hand (Left for the left controller and Right for the right controller) as shown in Figure 5.7.



**Figure 5.7: OVRHand Property Set Up to Each Hand**

**Step 7**:Create a new C# Script named **HandController.cs**. Right-click the **Create → C# Script** as shown in Figure 5.8.



**Figure 5.8: Create a Script Named HandController.cs**

**Step 8**: Open the C# Script and **copy the following code** into the **C#** Script named **HandController.cs.**

**Code Snippet 1:**

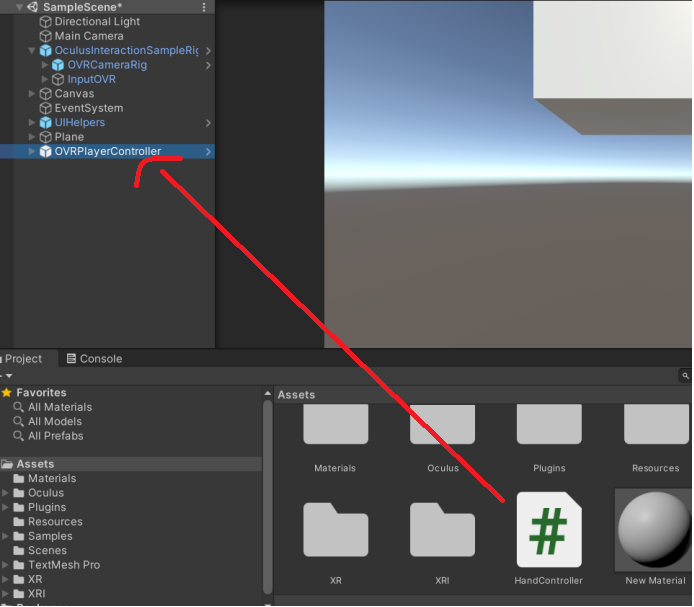
|  |
| --- |
| using UnityEngine;  using System.Collections;  using OculusSampleFramework;  public class HandController : MonoBehaviour  {  public OVRHand leftHand;  public OVRHand rightHand;  void Start()  {  leftHand = GameObject.Find("OVRPlayerController/Hands/HandLeft").GetComponent<OVRHand>();  rightHand = GameObject.Find("OVRPlayerController/Hands/HandRight").GetComponent<OVRHand>();  }  void Update()  {  // Check if the left hand is tracking  if (leftHand.IsTracked)  {  // Update the position and rotation of the left hand object to match the left controller  transform.Find("LeftHand").position = leftHand.transform.position;  transform.Find("LeftHand").rotation = leftHand.transform.rotation;  }  // Check if the right hand is tracking  if (rightHand.IsTracked)  {  // Update the position and rotation of the right hand object to match the right controller  transform.Find("RightHand").position = rightHand.transform.position;  transform.Find("RightHand").rotation = rightHand.transform.rotation;  }  }  } |

This code assumes that you have created two empty game objects in your scene called **LeftHand and RightHand**, which will be used to represent the virtual hands in the VR environment.

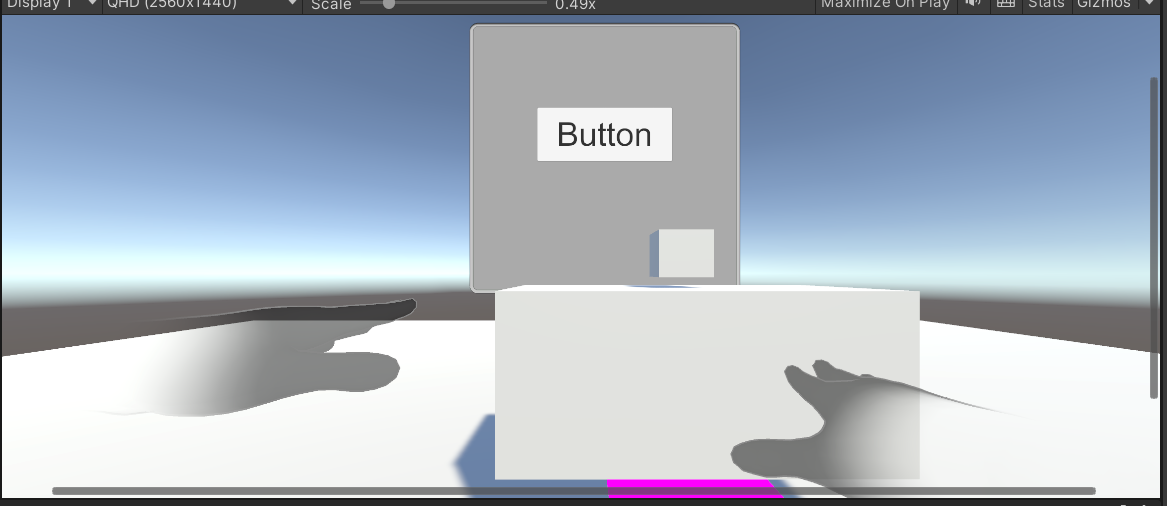
In the **Start** method of the script, the GetComponent method is used to get a reference to the **OVRHand** components attached to the left and right controllers.

In the **Update** method of the script, it is checked if each OVRHand component is tracking a hand. If it is, the position is updated and rotation of the corresponding virtual hand object to match the position and rotation of the controller.

**Step 9:** Attach script named **HandController.cs** to **OVRPlayercontroller** in Hierarchy Panel as shown in Figure 5.9.

**Figure 5.9: Attach the HandController Script**

**Step 11**:Test the app on **Oculus HMD**.Figure 5.10 shows final output.



**Figure 5.10: Final Output**

5.4 Summary

* **OVRPlayerController** is a Unity script that provides basic player movement and input handling in VR using gamepad or keyboard input. It also supports teleportation and can be customized to adjust movement settings.
* **VR controller** is a device held in the user’s hand used to interact with virtual environments in VR applications, typically including buttons, triggers, and joysticks for actions such as grabbing and pointing.
* **OVRCameraRig** provides a convenient and customization way to set up the camera rig for a VR project in Unity.
* **OVRHand** is a component that tracks the movement of the user’s hands and fingers in VR environments, allowing developers to create more immersive experiences.
* **OVRHandPrefab** is a prefabricated hand model included in the Oculus Integration SDK for Unity that provides hand-tracking capabilities for VR applications. It can be customized and used with both Oculus Quest and Oculus Rift S controllers.

5.5 Check Your Progress



1. What are the functions of OVRCustomSkeletonPrefab?

a) Contains a custom skeleton hand model.

b) Contains a model of Hands Controller.

c) Contains a model of OVRController.

d) All of these.

1. What component in Oculus Interaction SDK is used for tracking the hand movements?
2. OVRHand
3. OVRHandController
4. OVRHandPrefab
5. OVRCameraRig
6. Which prefab provides a set of cameras that are used to render VR environment?
7. OVRCameraRig
8. OVRController
9. OVRHand
10. OVRVRender
11. What is the use of changing hands as VR Controllers?
12. Intuitive Control
13. Provides Accessibility
14. Faster Interaction
15. All
16. Which of these is not component of Unity Oculus Interaction SDK?
17. OVRCameraRig
18. OVRVRender
19. OVRPlayerController
20. OVRHandController

5.6 Answers

|  |  |
| --- | --- |
| 1 | a |
| 2 | a |
| 3 | a |
| 4 | d |
| 5 | b |

5.7 Try It Yourself

* Build your VR Hands Controller.
* Make movements using VR Hand Controller.