Session 7

Making an Object Distance Grabbable

Welcome to the session, **Making an object distance grabbable**.

This session illustrates how to make an object distance grabbable in Virtual Reality (VR).

In this session, students will learn to:

* Explain the importance of distance grabbing in VR
* Describe hand presence and grabbing
* Illustrate controlling input mapping

7.1 Importance of Distance Grabbing in VR

Distance grabbing, also known as long-range grabbing or remote grabbing, is a technique in VR that allows users to interact with and manipulate objects that are located at a distance from their physical position. In Unity, implementing distance grabbing functionality is crucial for providing a more immersive and intuitive VR experience. Here are some reasons why distance grabbing is important:

* **Enhanced immersion**:Grabbable objects in VR allow users to interact with the virtual environment more realistically, providing a sense of presence and immersion. It bridges the gap between the physical and virtual worlds.
* **Natural interaction**: Grabbing and manipulating objects with hand gestures closely mimic real-world actions. This intuitive interaction method makes it easier for users to understand and engage with the VR environment.
* **Spatial exploration**: Grabbable distance objects enable users to explore the virtual space more effectively. They can pick up and inspect objects from different angles, encouraging spatial awareness and facilitating exploration of the virtual environment.
* **Interaction variety**: Grabbable objects add diversity to the range of interactions available in VR experiences. Users can grab, move, throw, and manipulate objects, allowing for more dynamic and interactive gameplay or virtual simulations.
* **Cognitive engagement**: Grabbing and manipulating objects in VR can enhance cognitive engagement by requiring users to think spatially, solve puzzles, or perform physical tasks. This engagement can contribute to improved learning, problem-solving, and cognitive skills.
* **Multi-modal feedback**: Grabbable objects can provide haptic and visual feedback when interacted with, adding another layer of realism to the VR experience. This feedback helps to reinforce the user's perception and understanding of the virtual environment.
* **User agency and empowerment**: Enabling users to grab distant objects in VR provides a sense of agency and empowerment. It allows them to take control of their virtual surroundings, making them active participants in the experience rather than passive observers.

7.1.1 Grab Begin and Grab End

**Grab Begin** and **Grab End** are typically used to refer to events or functions related to object manipulation or grabbing in a virtual environment.

**Grab Begin**: It usually represents the start of a grab or interaction with a virtual object. It can be triggered when a user initiates a grab action, such as pressing a button or performing a specific gesture with a VR controller. When the **Grab Begin** is called, it typically activates the logic that enables the user to manipulate the object they are interacting with.

**Grab End**: It represents the end or release of a grab or interaction with a virtual object. It is triggered when a user stops gripping or holding the object, such as releasing a button or releasing their grip on the VR controller. When **Grab End** is called, it usually deactivates the grab logic and stops the manipulation of the object.

These events are commonly used in AR and VR applications to provide a more immersive and interactive experience by allowing users to interact with virtual objects as if they were physically present. By using Grab Begin and Grab End events, developers can implement functionality such as picking up and moving virtual objects, manipulating them, throwing them, or performing other actions based on the movements of user hand or controller input.

7.2 Player in OVRGrabber

In VR development using Unity, the **Player** and **OVRGrabber** are components commonly used in the Oculus VR framework, specifically designed for Oculus VR devices such as the Oculus Rift or Oculus Quest.

**Player**: The **Player** refers to the main character or avatar representing the user in the virtual environment. It typically consists of a controller or hand models that track the user's hand movements and input, allowing them to interact with the virtual world. The Player component handles input from the VR controllers, tracks their position and orientation, and provides functionality for interactions such as grabbing and manipulating objects.

**OVRGrabber:** The **OVRGrabber** is a component that works in conjunction with the Player to enable object grabbing and interaction. It is responsible for detecting and managing interactions between the Player's hands (or VR controllers) and virtual objects within the scene. The OVRGrabber component typically includes logic for detecting when the user's hand or controller is in proximity to an object, initiating the grab action, and providing functionality for manipulating the object while it is being held.

The OVRGrabber component provides features such as collision detection, object attachment, and physics simulation for realistic object interactions. It allows the user to grab virtual objects within the VR environment by pressing a button on the VR controller or performing a specific gesture, and then manipulate the position of the object and rotation based on the user's hand movements.

By utilizing the Player and OVRGrabber components in Unity, developers can create immersive VR experiences that allow users to interact with virtual objects in a natural and intuitive way, enhancing the sense of presence and engagement within the virtual environment.

7.2.1 Grip Transform Component

The **GripTransform** component refers to the transform component of the hand or controller object that represents the grip point for grabbing objects. It determines theposition and orientation from which objects are grabbed and held.

7.3 Steps to Make an Object Grabbable From a Distance

Developing **Distance Grab** in VR scene is explained in this section. Figure 7.1 shows an example of Distance Grab in VR Scene.

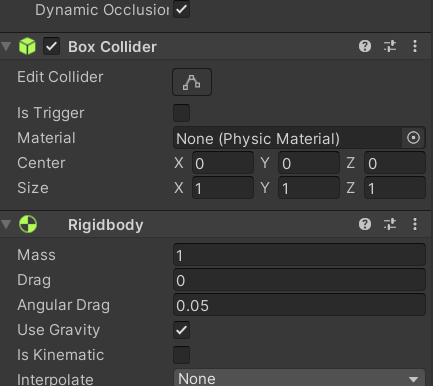
**Figure 7.1: Example for Distance Grab in VR Scene**

Following are the steps to achieve Distance Grab functionality in VR:

**Step 1**: Place the **OclusInteractionSampleRig** near the 3D Cube GameObject as shown in Figure 7.2.

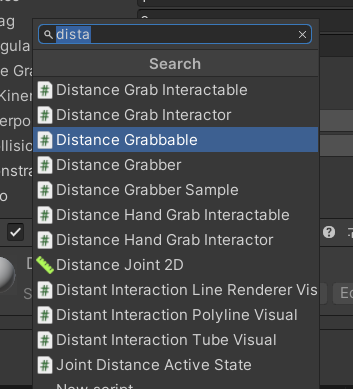
**Figure 7.2: OculusInteractionSampleRig is Dragged Near the 3D Cube Object**

**Step 2**:Ensure that theobject has a collider component attached to it. This will allow physics interactions. Figure 7.3 shows the 3D object with Box Collider and Rigidbody component.

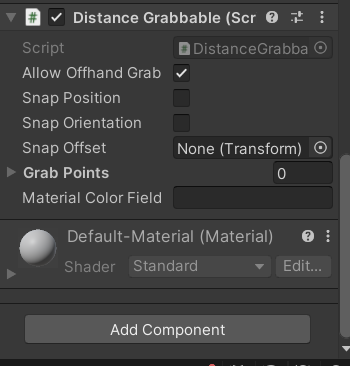


**Figure 7.3: Ensure 3D Object with Box Collider and Rigidbody Component**

**Step 3**:Select the Cube GameObject → Add Componentin the Inspector Panel and search for a script DistanceGrabbable.csas shown in Figure 7.4.

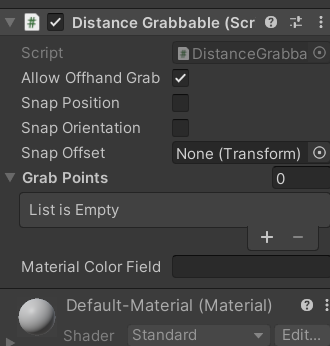
**Figure 7.4: Search the Script with Name DistanceGrabbable.cs**

**Step 4**: Select the script Distance Grabbable.csand click the Add Component button as shown in Figure 7.5.



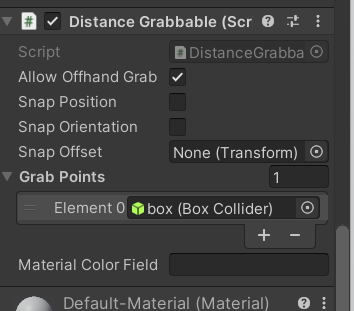
**Figure 7.5: Distance Grabbable Script Selected**

**Step 5**:Click the **Grab Points** and select **+** to make the object Rigid. Figure 7.6 shows that the Grab Points value is zero, initially.



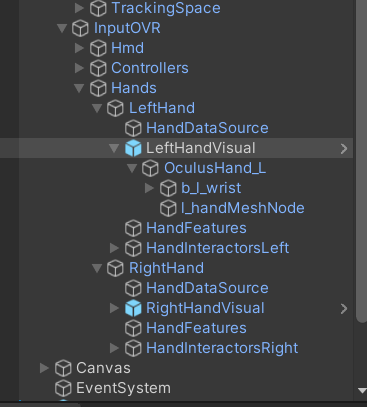
**Figure 7.6: DistanceGrabbable Script with No Grab Points**

**Step 6**:Drag **the Box Collider Component** to the **Grab Point Element**. Box Collider is displayed in Element 0 as shown in Figure 7.7.



**Figure 7.7: Adding Box Collider**

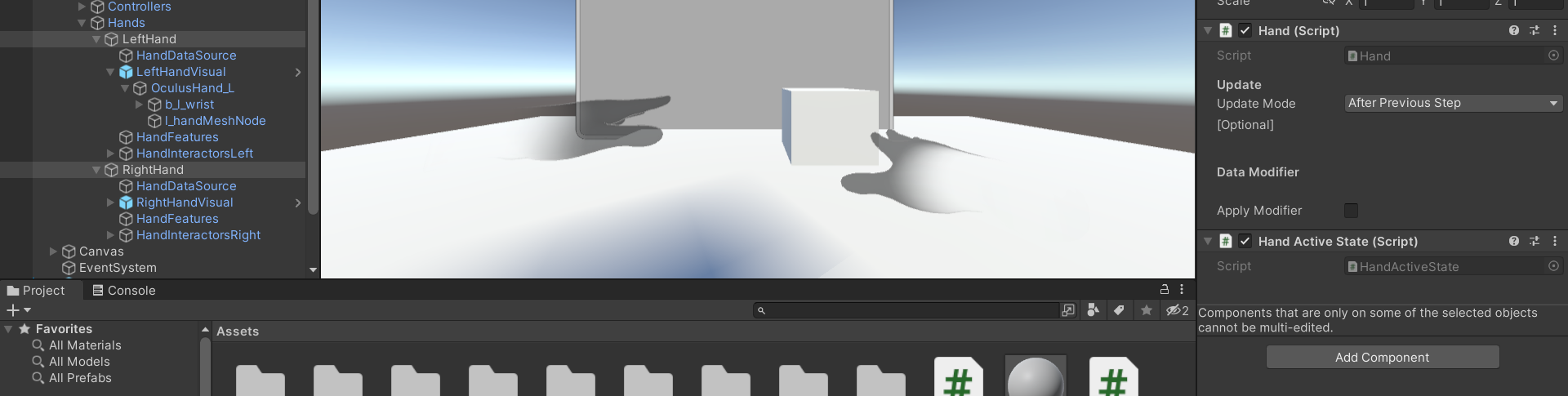
**Step 7**: In the Unity Editor's Hierarchy → select the hand model or controller object **which is Hands -> LeftHand -> LeftHandVisual in the Hierarchy Panel**  as shown in Figure 7.8.



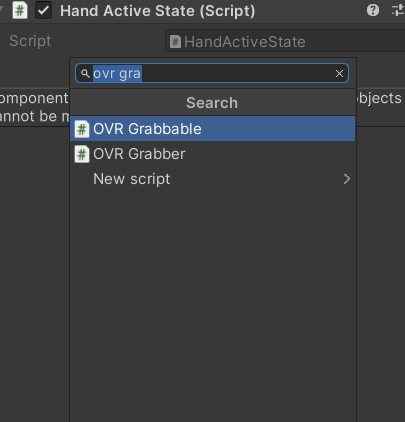
**Figure 7.8: Left Hand and Right Hand Controllers**

**Step 8**: Click the RightHand and Lefthand GameObject in Hierarchy Panel, select **Add component** in Inspector Panel, and search the script named **OVRGrabber.cs**.

Figure 7.9 shows LeftHand and RightHand Controller selected. Figure 7.10 shows searching **OVR Grabbable.cs** script.

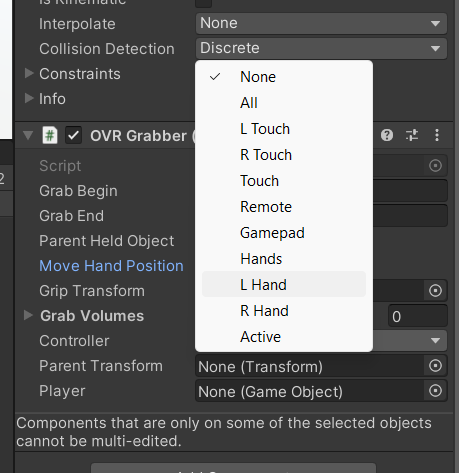


**Figure 7.9: Selecting LeftHand and RightHand Controllers**



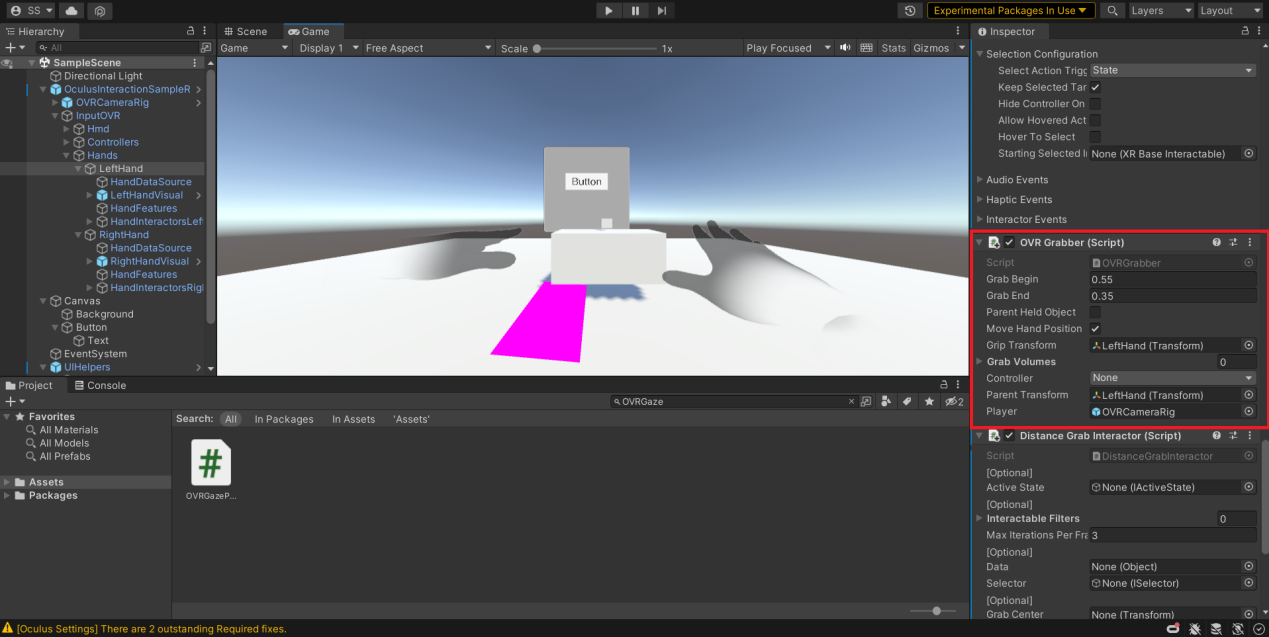
**Figure 7.10: OVR Grabbable.cs Script**

**Step 9**: Select **OVRGrabber** in Inspector and click the Grab Volumes → L Hand for **LeftHand GameObject** and R Hand for **Right Hand** GameObject. Set **Grab Begin** and **Grab End** as shown in Figure 7.11.



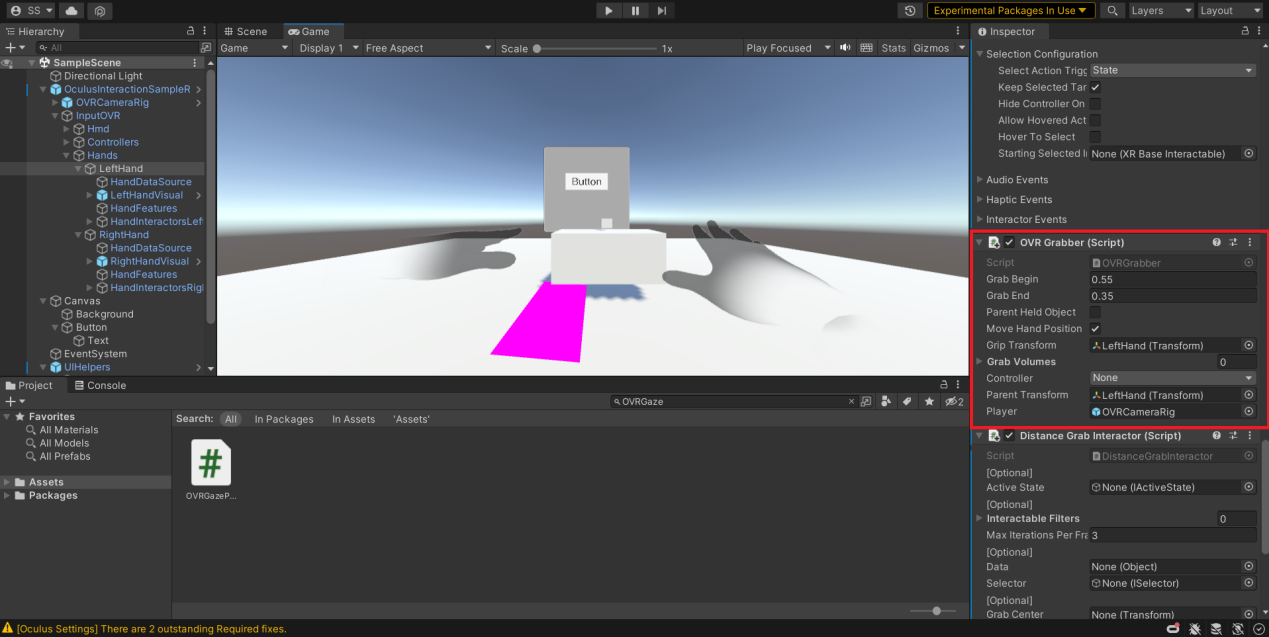
**Figure 7.11: Select Grab Volumes**

**Step 10**:Click the Parent Transform, **Grip Transform** and **Drag LeftHand GameObject** to Grip Transform as shown in Figure 7.12 do same for Right Hand as well.

****

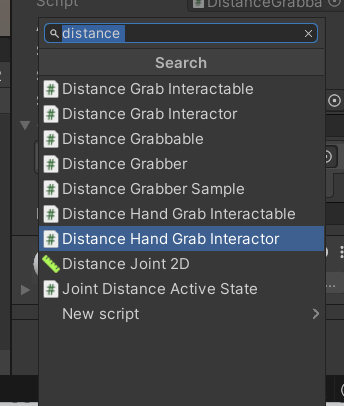
**Figure 7.12: Drag LeftHand and RightHand on Parent transform**

**Step 11**:In Player Inspector Panel, drag the **OVRCameraRig** component in Player as shown in Figure 7.13.

****

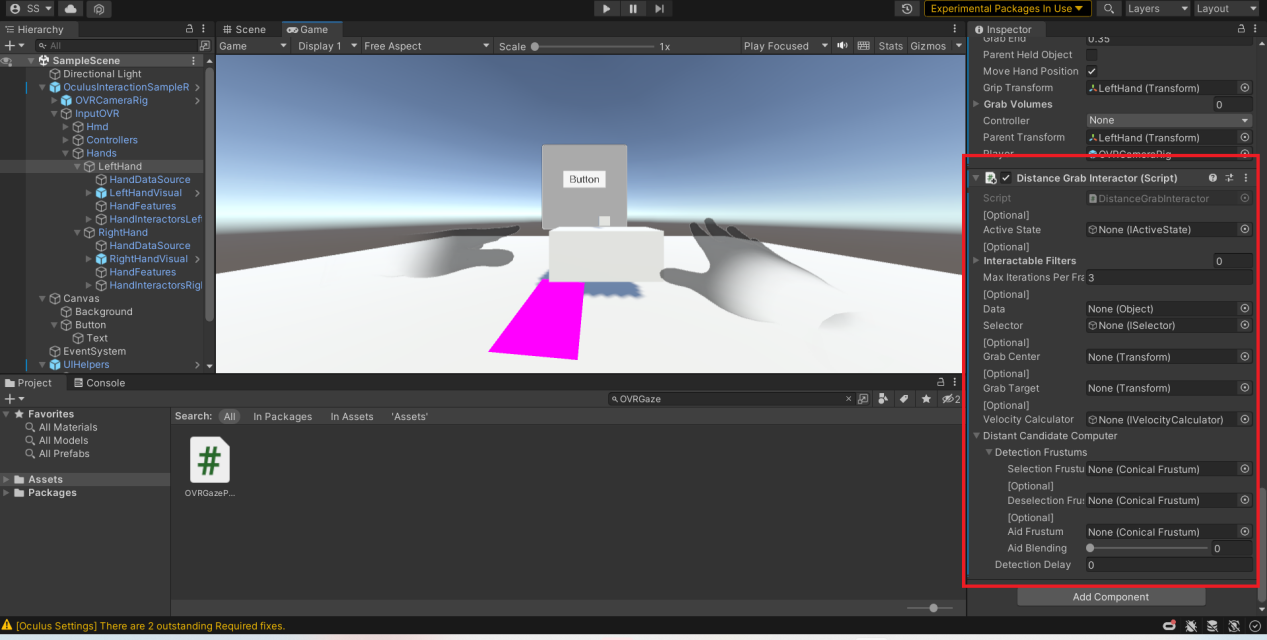
**Figure 7.13: OVRCameraRig is Added**

**Step 12**:Select Left and Right GameObject in the Hierarchy Panel and add a script with the name **DistanceHandGrabInteractor.cs** using the **Add Component** button as shown in Figure 7.14.



**Figure 7.14: Selecting Distance Hand Grab Interactor Script**

**Step 13**: **DistanceHandGrabInteractor.cs** script is designed to provide distance-based grabbing functionality. It would allow the player to interact with grab objects from a distance within a specified range as shown in Figure 7.15.

****

**Figure 7.15: DistanceHandGrabInteractor.cs Script is Added**

**Step 14**: Test the scene and verify that the object can be grabbed from a distance as shown in Figure 7.16.



**Figure 7.16: Final Output for Distance Grab**

This is how you make the object distance grabbable in the VR Scene.

7.3 Controlling Input Mapping

Controlling input mapping in VR in Unity refers to the process of configuring and mapping user input from VR devices, such as motion controllers or VR headsets, to specific actions or functions within the VR application. Unity provides various tools and methods to handle input mapping in VR.

**VR Input Systems**: Unity provides built-in VR input systems that help simplify input handling for VR applications. These systems include the Unity XR Input system, which offers a unified API for accessing input from different VR platforms (such as Oculus, HTC Vive, or Windows Mixed Reality). The XR Input system provides a common interface for accessing input devices, button states, and pose data from VR controllers.

**Input Mapping Setup**: Unity allows to define and customize input mappings for VR devices. This involves associating specific buttons, triggers, or gestures from the VR controllers with desired actions or functions in VR application. Unity provides an Input Manager where developer can create and manage input profiles for different VR devices, defining the button mappings and axis configurations.

**Input Handling Code**: Once the input mappings are set up, input must be handled within application's code. Unity provides different approaches for input handling, such as using events, polling input states, or utilizing Unity's Input System package. Code can be written that works for specific input events or queries the current state of input buttons or axes to trigger actions or manipulate objects within the VR application.

**Custom Input Handling**: Unity allows for customization of input handling to suit specific VR application requirements. Developer can define custom input logic, implement gesture recognition, or create complex interaction systems. This might involve tracking hand movements, detecting gestures such as swiping or pinching, or implementing physics-based object manipulation based on VR controller input.

**Testing and Iteration**: It is important to thoroughly test and iterate on input mapping implementation to ensure a smooth and intuitive user experience. Test the VR application with various VR devices, validate the input mappings, and gather user feedback to refine and optimize the input controls for comfort and usability.

By effectively controlling input mapping in VR using Unity, developer can create immersive and interactive VR experiences that respond accurately to user actions and provide an intuitive means of interaction within the virtual environment.

7.4 Summary

* LeftHand and RightHand in VR refer to the virtual representations or controllers associated with the user's left and right hands, respectively. They allow natural interaction and manipulation of objects in the virtual environment.
* Distance Grab in VR enables users to interact with objects and manipulate them from a distance, extending their virtual reach for intuitive and immersive interactions.
* DistanceGrabbable.cs is a script in the Oculus Integration SDK that allows objects to be grabbed and manipulated in VR from a Distance.
* OVRGrabber.cs is a script in the Oculus Integration SDK that enables hand or controller grabbing functionality in VR.
* DistanceHandGrabInteractor.cs script is designed to provide distance-based grabbing functionality.

7.5 Check Your Progress



1. Why to use Distance Grab in Virtual Reality Software?
2. Enhanced immersion
3. Spatial exploration
4. Interaction variety
5. All of these
6. Which script allows objects to be grabbed and manipulated in VR?
7. DistanceGrabber
8. OVRGrabbable
9. OVRGrabber
10. ObjectGrab
11. OVRGrabber is a script that enables hand or controller grabbing functionality in VR.
12. OVRGrabber
13. OVRGrabbable
14. OVRGrab
15. OVRTakeObject
16. Which GameObject goes into Player Component of OVRGrabbable Component?
17. OVRCameraRig
18. OVRCamera
19. GameObject of Player
20. OVRPlayer

5.Which component is important for object physics to manipulate and grab a GameObject?

1. Rigidbody
2. Box Collider
3. Both a and b
4. None of these

7.5.1 Answers

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

7.6 Try It Yourself

* Build a distance grab system.
* Test the Application in VR Headset.