Tutorial locomotions and Interactions:

1. Linear Locomotion

Step 1:

The linear locomotion techniques basically performs linear translational and rotational movements. It can be easily achieved by the Oculus integration SDK without coding. Let's get started! First, please download the **Oculus integration SDK In the asset store:**

♥ <> Ø ■ I # ×	Center Global 13		n Personal (Personal) «Metal>	Collab -	Account - Layers - Layout
Asset Store 🗢 Game				'≡ Hierarchy 🔒 :	O Inspector 🔀 Navigation
♦ P A Categories •	Sell Assets Feedback FAQ 🚯 My Assets		Open in Browser English - USD -	+ • (9: All	OVRPlayerController (Prefab
				▼ BasicLocomotion* :	
😴 unity Ass	set Store Search for assets	Q ± ♥ ₽		Main Camera Oriectional Light	
				► OVRPlayerControlle >	Open Prefab
				DistanceGrabHand >	
	RECENT SEARCH			Interactable objects Walk in place	Open Prefab for full editing support.
	Oculus Integration		clear filters	Q	
					Root in Prefab Asset
	oculus	ories			
	oculus	onea			OVRPlayerController Sta
		(713)	~		Tag Untagged + Layer Default
					▼ 🙏 Transform 🛛 🖗 🛱
1		158)	~		Position X 0 Y 0 Z 0
	O	ns (34)	~		Rotation X 0 Y 0 Z 0
	Gr	113 (34)	Ť		Scale X 1 Y 1 Z 1
		(5548)	~		▼ (0) ✓ Character Controller @ #
and the second	VO				
and the second second	yU	tials (42)	~		Slope Limit 45 Step Offset 0.3
	Ending	(070.4)	~		Skin Width 0.01
	Ending	ates (2724)	Ť		Min Move Distance 0
\square		(8200)	~		Center
					X 0 Y 0 Z 0
and the second		VFX (2339)	~		Radius 0.5
Scene Project			<u> </u>		Height 2
 editor 	Assets > Oculus > VR > Prefabs	٩	♣ ♥ ★ ∞14		🔻 🗯 🗹 OVR Player Controller (: 🐵 📫
Plugins	Assets > Oculus > VR > Pretabs Gursor_Timer				Script # OVRPlayerCo
Im scenes	OVRCameraRig				Acceleration 0.1
▶ Im scripts ▼ Im VR	OVRControllerPrefab				Damping 0.3
AudioClips	OVRCubemapCaptureProbe OVRCustomHandPrefab_L				Back And Side Damp 0.5
Editor Materials	OVRCustomHandPrefab_R				Jump Force 0.3 Rotation Amount 1.5
▶ Meshes	OVRHandPrefab OVRPlayerController				Rotation Ratchet 45
Plugins					Snap Rotation
Prefabs Resources					Fixed Speed Steps 0
Scenes					Hmd Resets Y
Scripts					Hmd Rotates Y
Shaders Textures					Gravity Modifier 1
🖿 ThirdParty					Use Profile Data 🗸
Resources Scenes					Enable Linear Moverr
V Car XR			•		Asset Labels
No new spatializer plugin(s) found					Auto Generate Lighting (

Step 2: Create an empty scene with the + button from the project panel. Name whatever you want.

	iotion - LocomotionAndinteractions - Android - Unity	2019.3.111 Personal (Personal) <metal></metal>	
🕊 😔 🕼 🗵 🖽 🛠 🖉 Center 🤀 Global 🖽			Collab 🔻 📥 Account 👻 L
Asset Store 🕫 Game		'≡ Hierarchy 🔒 🗄	Inspector Navigation
Free Aspect Scale 1x	Maximize On Play Mute Audio Stats Gizmos 🔻	+ - Q: All	BasicLocomotion
÷			4
# Scene In Project	≗ :		
Folder Assets > Scenes CH Script Stader Stader Stader Testing > Payables Assembly Definition Assembly Definition > Assembly Definition > Schne ✓ Basic Locomotion ✓ Schne ✓ Schne ✓ Audio Mixer Niers Material Lens Flare Render Texture Lightmap Parameters Custom Render Texture Animator Controller Animator Controller Animator Controller			Accest above

Step 3: In the hierarchy panel, use the + button to create the floor and place some object as indicators in the scene. You can be creative.

		Stion - LocomotionAndinteractions - Android - Onity		
👋 💠 💭 🔝 🔛 🋞 🛠 🖉 Center 🌐 Global 🗄		► II ►		Collab 🔻 📥 Account 🔻 L
Asset Store 🛛 👁 Game			'≡ Hierarchy a :	Inspector X Navigation
Free Aspect Scale 1x		Maximize On Play Mute Audio Stats Gizmos 🔻		BasicLocomotion
		Maximize On Play Mute Audio Stats Gizmos V Maximize On Play Mute	Image: Second	Inspector X Navigation BasicLocomotion
Editor Mage Viage Viage Viage Viage Spatializer Medior Mage Sones Mage Viage Name Sones Sones Socees Socees Socees Viage Via				
Loaders Packages G Assets/Scenes/B				Asset Labels
► Foregoe	NICLOCOMOTION UNITY			Asset Lauels

	Te	BasicLocomotion - LocomotionAndinteractions - Android - Ul BasicLocomotion - LocomotionAndinteractions - Android - Ul	nity 2019.3.111 Personal (Personal) <metal></metal>	
🖤 🕂 🖸 🖻 🗶 🖉	OCenter Global 5	► II ►		Collab - Account -
Asset Store 🗢 Game				Inspector 🔀 Navigation
ree Aspect 🔹 Scale 🖝	1x	Maximize On Play Mute Audio Stats Gizmos		
			Create Empty tion	
			oroate empty orma	>
			3D Object Cube 2D Object Sphere	>
		č -		Ś
		×	Light > Cylinder	
			Audio 🕨 Plane	
			Video 🕨 Quad	
			UI Text - TextMeshPro Camera ProBuilder Cube	
			Camera ProBuilder Cube Ragdoll	
			Terrain	
			Tree	
			Wind Zone	
			3D Text	
# Scene Project		<u>۵</u>	1	
		<u>م</u>		
Audio	Assets > Scenes			
Materials	BasicLocomotion			
Prefabs Scripts	SampleScene			
Schpis Shaders	C Teleportation			
V Decomotion				
Materials				
Models				
Scripts				
TouchControllers Animations				
Materials				
Models				
NewTouchControllers				
Video				
Editor				
 Usage Spatializer 				
spatializer editor				
Plugins				
Image: Scenes				
scripts				
▶ m VR				
Resources				
Scenes 5				
V DA XR				
V C XR				
V Dates	- @ Assets/Scenes/Basicl.ocom	tion unity		
Loaders	Assets/Scenes/BasicI ocom	tion unity		

Step 4: If we check the Oculus SDK package-> VR - > Prefabs: You can found the prefabs that we can use to create basic interactions and locomotions. The most basic two are OVRCameraRig and OVRPlayerController.

+ -	Q	* 🖉 ★ 💋 1
 Favorites All Materials All Models All Prefabs Assets Coulus Avatar LipSync Platform SampleFramework Spatializer VR AudioClips Editor Materials Meshes Plugins Prefabs Resources Scenes Scripts Shaders Textures 	 Assets > Oculus > VR > Prefabs Cursor_Timer OVRCameraRig OVRCubemapCaptureProbe OVRCustomHandPrefab_L OVRCustomHandPrefab_R OVRPlayerController 	▶ ♥ ★ ⊅ 1
ThirdParty Resources		
	Assets/Oculus/VR/Prefabs/OVRPlayerController.prefab	•

OVRCameraRig

is a custom VR camera that may be used to replace the regular Unity Camera in a scene. Drag an OVRCameraRig into your scene and you will be able to start viewing the scene. The primary benefit to using OVRCameraRig is that it provides access to OVRManager, which provides the main interface to the VR hardware. For example, in the inspector, we can check if the device is Quest or Go.

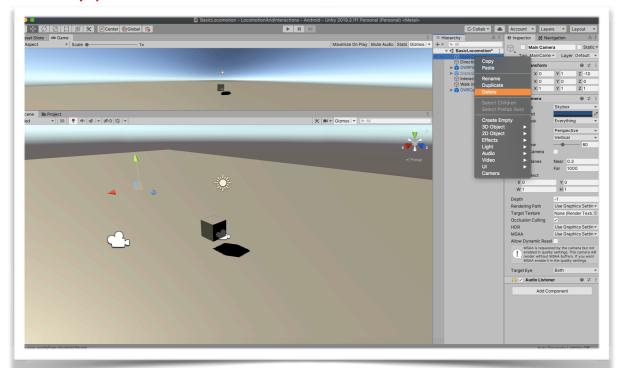
🔻 # 🗹 OVR Camera Rig (So	cript)	Ø	-1-	:
Script	OVRCameraRig			۲
Use Per Eye Cameras				
Use Fixed Update For Tracki				
Disable Eye Anchor Camera	s			
🔻 # 🗹 OVR Manager (Scrip	it)	0		:
Target Devices				
Gear Vr Or Go				
Quest	\checkmark			
Script	# OVRManager			۲
Performance/Quality				
Queue Ahead	~			
Use Recommended MSAA L	•			
Monoscopic				
Enable Adaptive Resolution				
Min Render Scale	•	- 0).7	
Max Render Scale	•	- 1		
Head Pose Relative Offset R	(X 0 Y 0 Z	0		
Head Pose Relative Offset T	IX 0 Y 0 Z	0		
Profiler Tcp Port	32419			
Tracking				
Tracking Origin Type	Eye Level			•
Use Position Tracking	~			
Use IPD In Position Tracking	~			
Reset Tracker On Load				
Allow Recenter	~			
Reorient HMD On Controller	~			
Late Controller Update	✓			
Quest Features				
Focus Aware				
Color Gamut	Default			•
Hand Tracking Support	Controllers Only			-

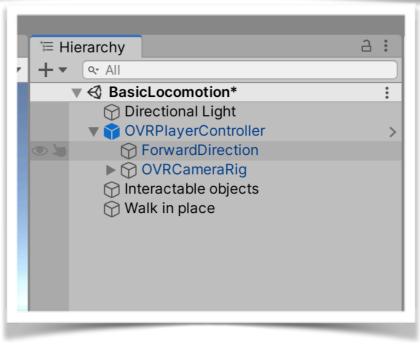
OVRPlayerController

The OVRPlayerController is the easiest way to start navigating a virtual environment. It is **basically an OVRCameraRig prefab attached to a simple character controller.** It includes a physics capsule, a movement system, a simple menu system with stereo rendering of text fields, and a cross-hair component. Hence, we can simply drag the prefab in the scene, and move around with the controllers.

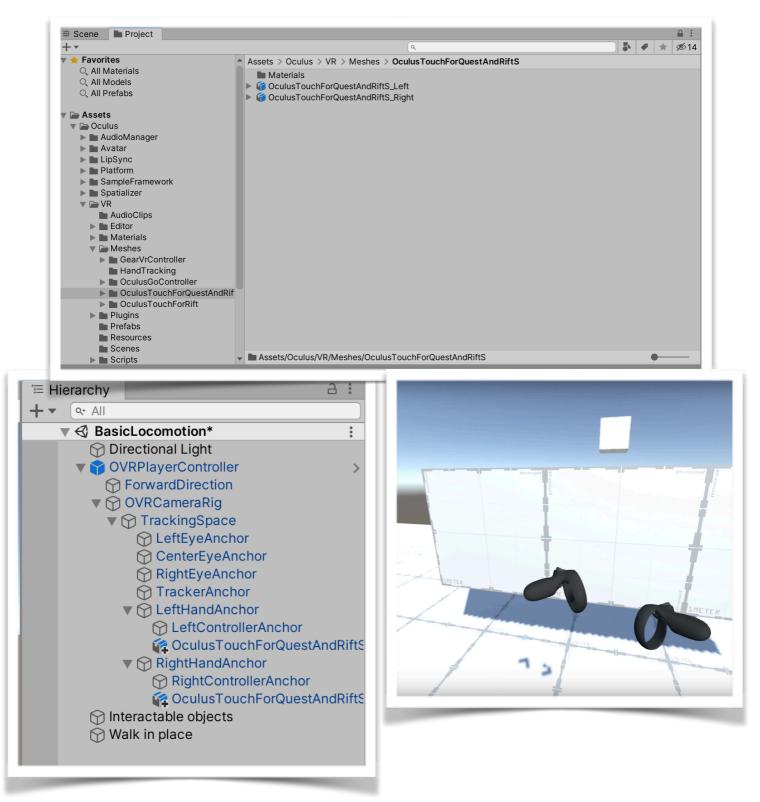
Note: Since the OVRPlayerController already have a camera with it. You can delete the original MainCamera in the scene.

Make sure you only have one camera in the scene(Unless you have other usages like 2D mini maps)





Step 5: To make the oculus quest controllers visible in the scene, You can add the model prefabs attached to the Left and right hand anchor as a Child. See the pictures below:



Now, you should be all set!!!

The right controller is to turn and the left one is to move.



Step 6: Finally, let's take a closer look at the details in the main script: OVRPlayerController. There're a lot of parameters you can modify and try freely. One specific parameter we mentioned here is the SnapRotation.

 ▼ # ✓ OVR Player Controller (Script) Ø - ÷ 			:	
Script	# OVRPlayerController			\odot
Acceleration	0.1			
Damping	0.3			
Back And Side Dampen	0.5			
Jump Force	0.3			
Rotation Amount	1.5			
Rotation Ratchet	45			
Snap Rotation	\checkmark			
Fixed Speed Steps	0			
Hmd Resets Y	~			
Hmd Rotates Y	✓			
Gravity Modifier	1			
Use Profile Data	\checkmark			
Enable Linear Movement	\checkmark			
Enable Rotation	\checkmark			
Rotation Either Thumbstick				

With this parameter toggled off. You will have the Smooth Turn, which could offer you more detailed information of the scene during turning. However, to user who is sensitive of cybersickness, SnapTurn is much better than the Smooth Turn. With this parameter toggled on, you can have the so-called Snap Turn, Similar to the next locomotion techniques that we will implement, it has a discrete nature. Try to modify the RotationRatchet Parameter to play.

Open the script OVRPlayerController.cs. The code on this capture is the section handling the snap rotation set on the previous picture. The RotationRatchet Parameter defines the degree that you will turn each time you press the button.

430		
		ID UNITY_EDITOR
		!SkipMouseRotation)
		euler.y += Input.GetAxis("Mouse X") * rotateInfluence * 3.25f;
	#endif	
	if (SnapRotation)
	{	
		if (OVRInput.Get(OVRInput.Button.SecondaryThumbstickLeft)
		(RotationEitherThumbstick & OVRInput.Get(OVRInput.Button.PrimaryThumbstickLeft)))
		{
		if (ReadyToSnapTurn)
		This parameter is defined in the beginning of
		(euler.y -= RotationRatchet;) this period on a floot. You can adjust the
		number to see the changes.
		else_if (OVRInput.Get(OVRInput.Button.SecondaryThumbstickRight)
		(RotationEitherThumbstick & 0VRInput.Get(ÓVRInpi 🛛 OVRInput.Button.SecondaryThumbstickRight = 0x08000000
		{
		if (ReadyToSnapTurn)
		euler_y += RotationRatchet;
		ReadyToSnapTurn = false;
		else
		t ReadyToSnapTurn = true;
	}	Γ
	else	
	ì	<pre>Vector2 secondaryAxis = 0VRInput.Get(0VRInput.Axis2D.SecondaryThumbstick);</pre>
		if (RotationEitherThumbstick)
		<pre>Vector2 altSecondaryAxis = 0VRInput.Get(0VRInput.Axis2D.PrimaryThumbstick);</pre>
		if (secondaryAxis.sgrMagnitude < altSecondaryAxis.sgrMagnitude)
		{
		secondaryAxis = altSecondaryAxis;
		}
		}
		euler.y += secondaryAxis.x * rotateInfluence;
	}	
	tran	sform.rotation = Quaternion.Euler(euler);
470	1	