Setting Up STEAMVR and Leap Motion

Playing nice with others

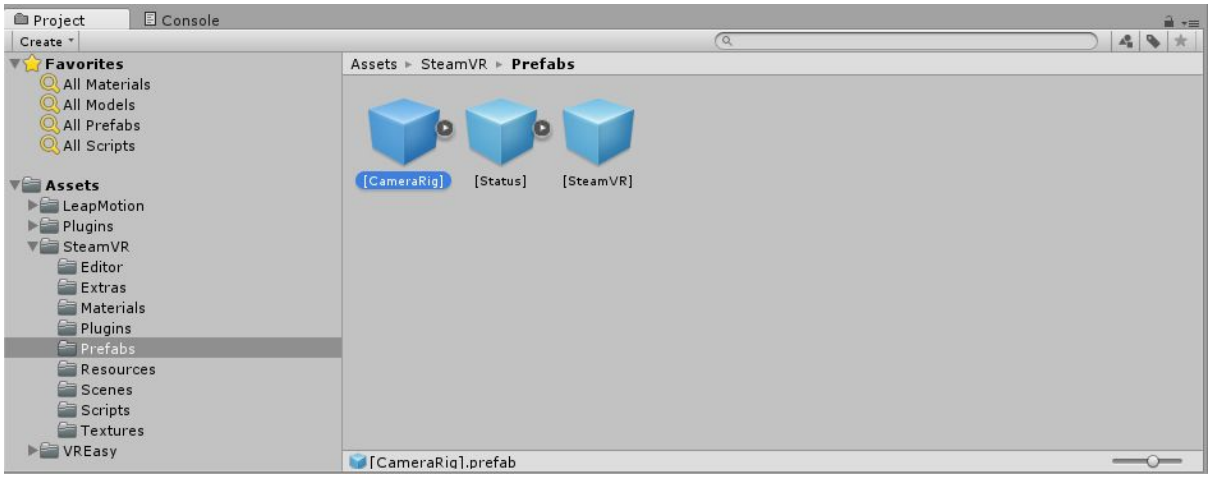
Adding VREasy functionality to established VR assets is easy. Here we comment on the most popular choices.

STEAMVR plugin

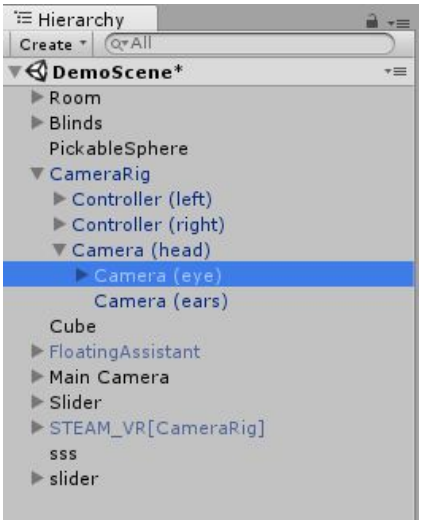
There are three common ways to make VREasy interact with SteamVR plugin: 1) Adding sight selection to their camera / HMD prefab; 2) Adding touch selection to their controller and grab functionality; 3) Using their controller as input selection for locomotion

Adding sight selection to their camera

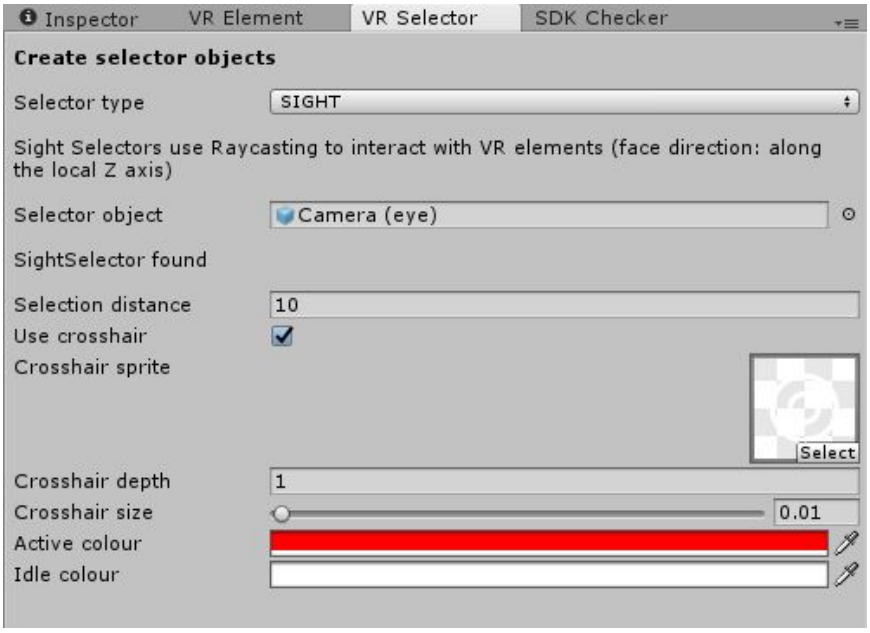
1) Drag their CameraRig prefab to your scene



2) Find the gameobject within the CameraRig hierarchy that represents the eye-sight of the HMD. In the case of CameraRig, this is the child object Camera(eye) that can be found by navigating CameraRig > Camera (head) > Camera (eye)



3) Open the VR Selector menu (VREasy > VR Selector) and choose Sight in the Selector type field. Drag and drop the Camera (eye) game object to the Selector object field.
4) Click Add selector and the system will add the functionality to it. You can now move on to configuring the SightSelector as you would normally do. In this example, I have added a crosshair and left all the properties by default

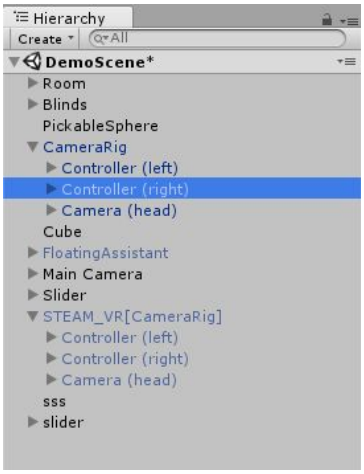


5) You are done! Now the Steam VR HMD object should be able to interact with VR Elements

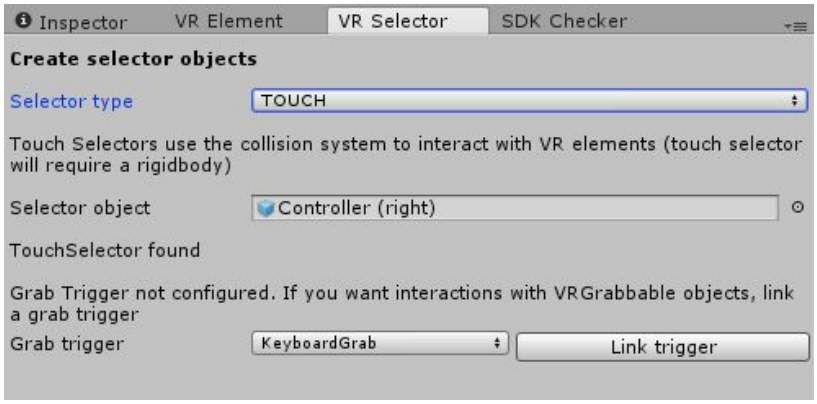
Adding touch selection to their controller

The process to add a touch selector to a steam controller is similar to that of a sight selector.

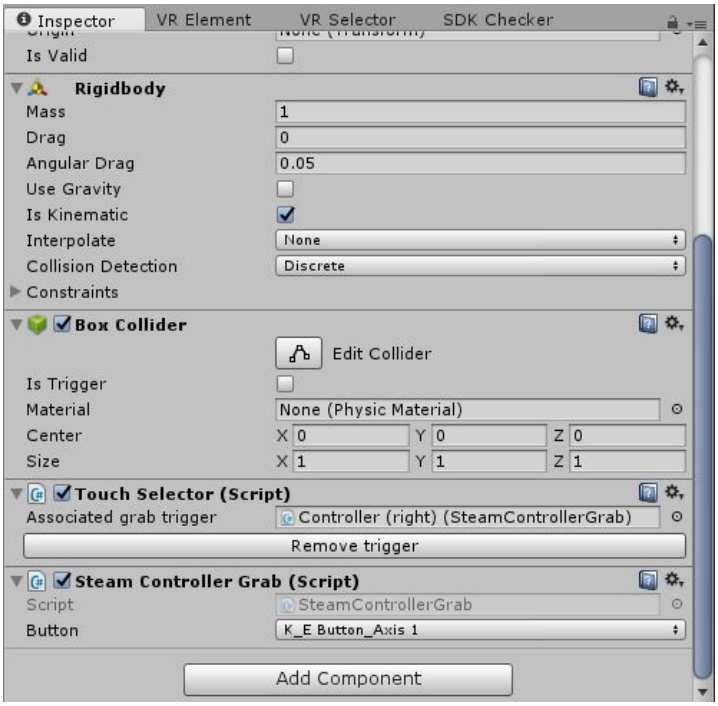
1) Find the Steam CameraRig prefab in your scene (if you don't have one, add one). Select the controller child that you want to convert to a touch selector; this will be either Controller (left) or Controller (right). If your project does not make use of the steam CameraRig prefab but you have any other object that is controlled by the steam vr controller, use that object instead.



2) Open the VR Selector menu (VREasy > VR Selector) and choose Touch as your Selector type.
3) Drag the object in step 1 to the Selector object field and click on Add TouchSelector



4) If you want your touch selector just for selecting VR elements, then you are done! If you wish to also grab grabbable objects, then select any of the Grab triggers in the VR Selector menu and link it. Since you are using the Steam VR controller for touching and selecting, it makes sense to also use it as input to start / stop grabbing objects ☺ For that, select SteamControllerGrab as a Grab trigger and click on Link trigger
5) Now select your object in step 1 in the hierarchy and configure it in the inspector. The component you are looking for is called SteamControllerGrab and it contains a list of all the buttons in the SteamVR controller. Select whichever you prefer -by default, the Axis 1 is selected, which corresponds to the analogue trigger.

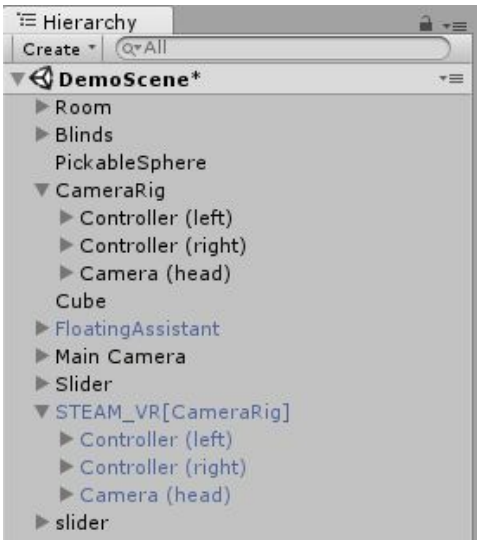


NOTE: By default the VR Selector menu will add a BoxCollider if no collider was found in the Controller object. This BoxCollider is a standard 3D box with 0.1 as width, depth and height and will probably suit most scenarios. However, if a different type of collider is wanted, or different dimensions are required, please add the specific collider to your Controller gameobject before step 2.

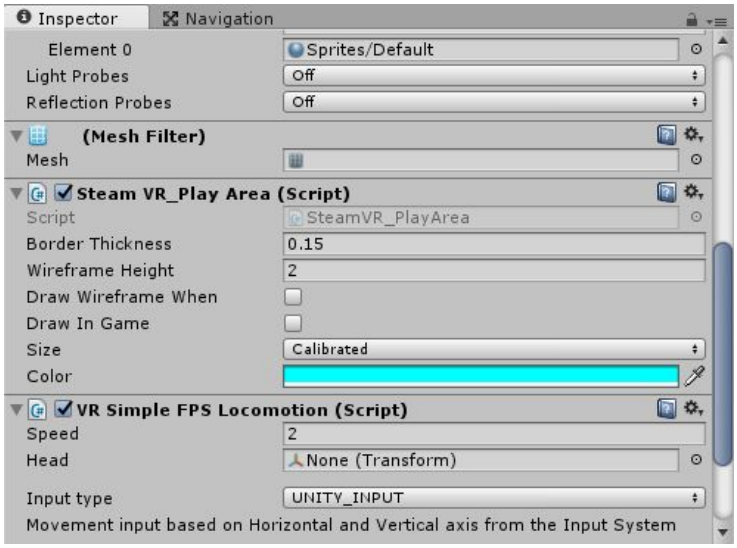
Using Steam Controller as input for the locomotion script

This script is more of a demo, but if you find it useful you can of course use it. To use it with the Steam CameraRig prefab, please do the following:

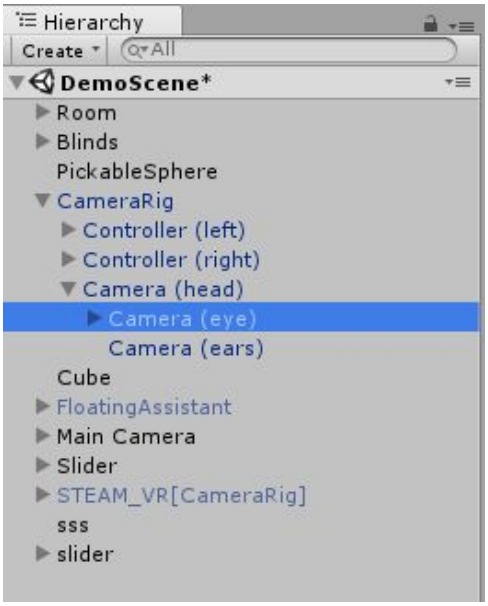
1) Find the Steam CameraRig object in your scene; if you do not have one, drag it from the SteamVR/Prefabs folder
2) Drag the VRSimpleFPSLocomotion script to the root of the game object



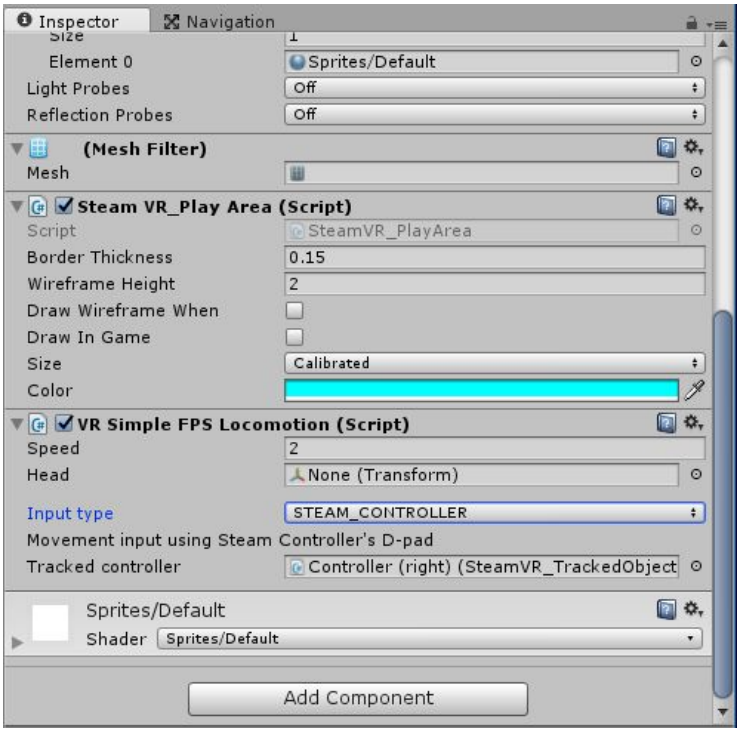
3) Select that game object and configure the VRSimpleFPSLocomotion script in the inspector



4) There are two properties that must be set. The first one is the Head, which indicates what game object in your scene represents the HMD, or the head of the player. If using the Steam CameraRig, this is the child game object called Camera (eye) that can be found on CameraRig > Camera (head) > Camera (eye). Drag the Camera (eye) transform to the Head field.



5) The second property to set is the Input type you wish to use to control your movements. There are two options: Unity_input (will use the Input axis “Horizontal” and “Vertical” to control the FPS camera) and Steam_Controller (evidently it will use the steam controller). If you select Steam_Controller, you will have to drag the specific controller (SteamVR_TrackedObject) that you wish to use to the Tracked controller field. If for instance you wish to use the right hand controller, drag the Controller (right) game object here.



You are ready to go! Control movement by touching the D-pad in the Steam controller.

LEAP MOTION

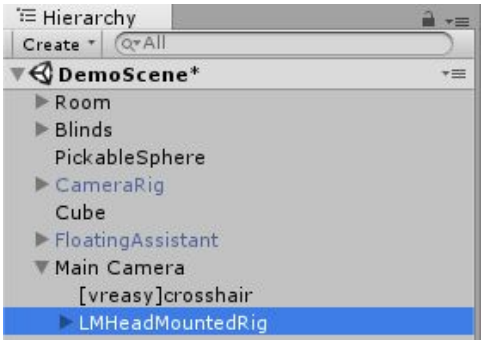
The instructions laid out here are specific for the Orion Leap Motion -which is designed specifically to play nice with VR headsets. I have used the version of the Core Assets 4.1.4. Note that it is also possible to make them work with previous versions of the Leap Motion Unity Asset, but those require you to find the appropriate Leap Motion Physics hand model child object and use it as VR Selector.

For the demo I have created a prefab out of their Leap_Hands_Demo_VR scene -for some reason their basic prefabs are not fully configured to just drag and drop. So this is a sort of pre step:

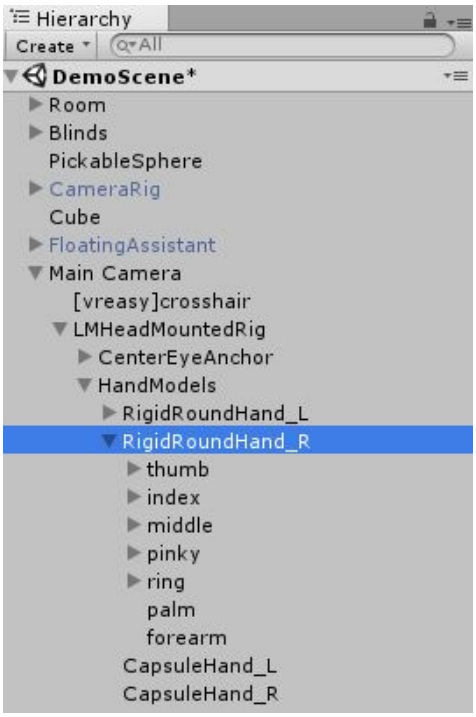
1) Once you have imported the Leap Motion Orion Core Assets, open their Leap_Hands_demo_VR scene. Make a prefab of their LMHeadMountedRig game object. You can place it anywhere in your project folders.

To convert your Leap Motion controllers (i.e. your hands) to VR Selectors, you have to use the VR Selector GUI and follow these steps:

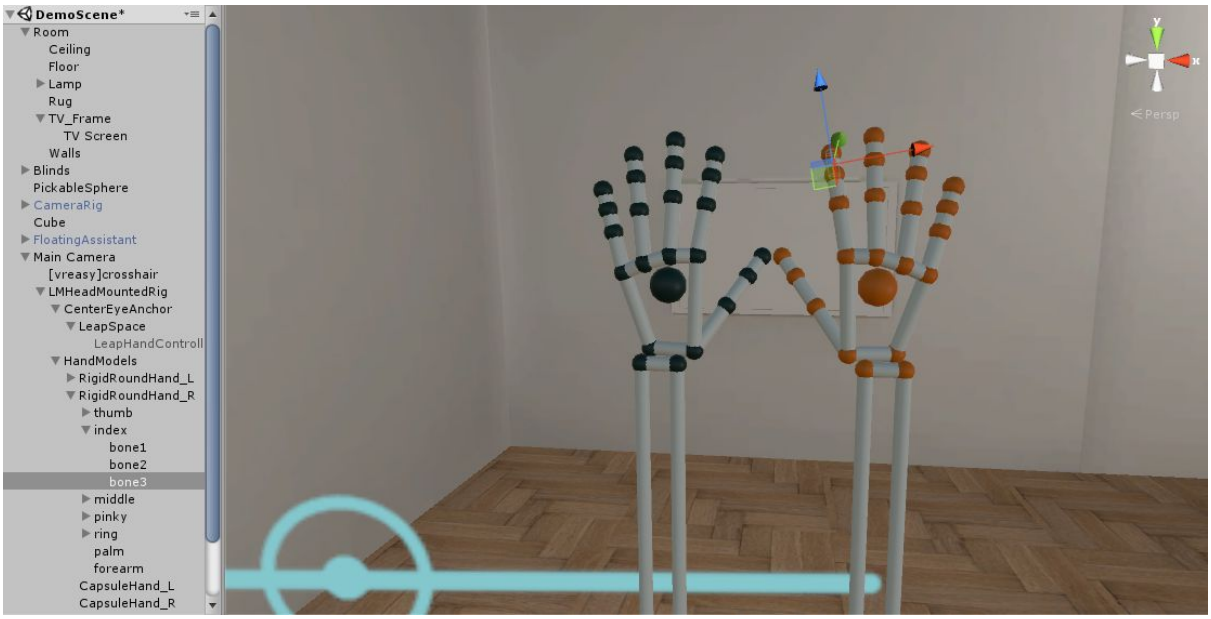
1) Drag the LMHeadMountedRig to your scene and make sure it is a child of your Main Camera. If you are using the default Main Camera from Unity as a HMD, make that Main Camera the parent. If you are using HTC Vive prefab for the HMD, Use their Steam_VR[CameraRig] as the parent.



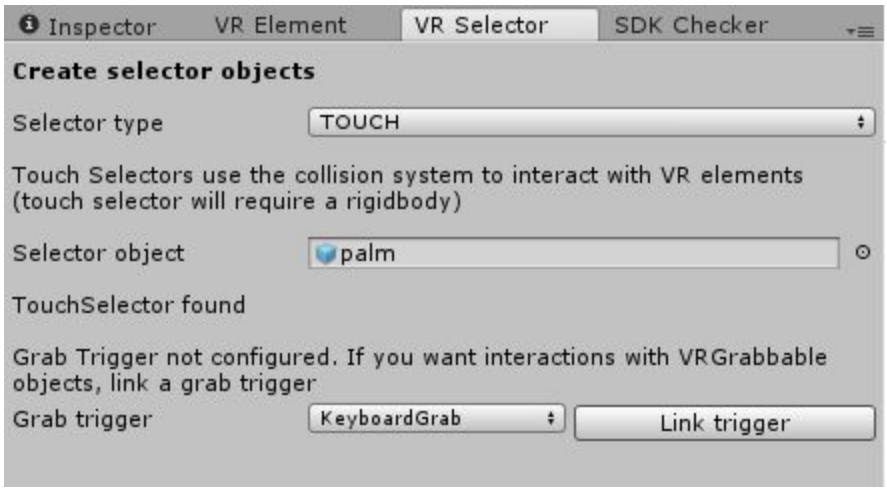
2) You should be able to play and see the Leap Hands appearing in front of your HMD when you wave at it (make sure the physical LM is actually present!)
3) Open VR Selector GUI and select Touch as selector type
4) Now this is the tricky bit! A LM hand has a physical representation that is made up of several colliders in Unity (roughly speaking, one per bone in your hand). You can use ANY of those bones as a VR touch selector, but you can also use ALL of them -for that you will have to repeat steps 5 to 7 for each bone). In this demo we will make the palm of your right hand a VR Selector -much easier to aim than using a single finger bone.
5) Find the bone representation you wish to use. Open the LMHeadMountedRig and find the physical model object you want to use (if you are using your own built Leap Motion hand, the names will vary, but the essence will be the same); in our case, it is the RigidRoundHand_R.



6) Open to display the children objects. You should see a game object per finger, plus palm and forearm representations. From this hierarchy you can use any of these as selectors, **provided** they have a rigidbody and a collider. If you want to use the palm and the forearm, those are ready to use. If you want to use the index finger, you have to be more specific and select one of the children bones (as each finger will have multiple bones). Again, you can use as many as you want as selectors, but to be able to use VREasy with LM you will have to choose at least one. In this case, we will select the palm object. Note that you can see in the editor window which part of the hand this will be; you will be able to select and touch VR elements *only* via the bones you make VR Selectors.



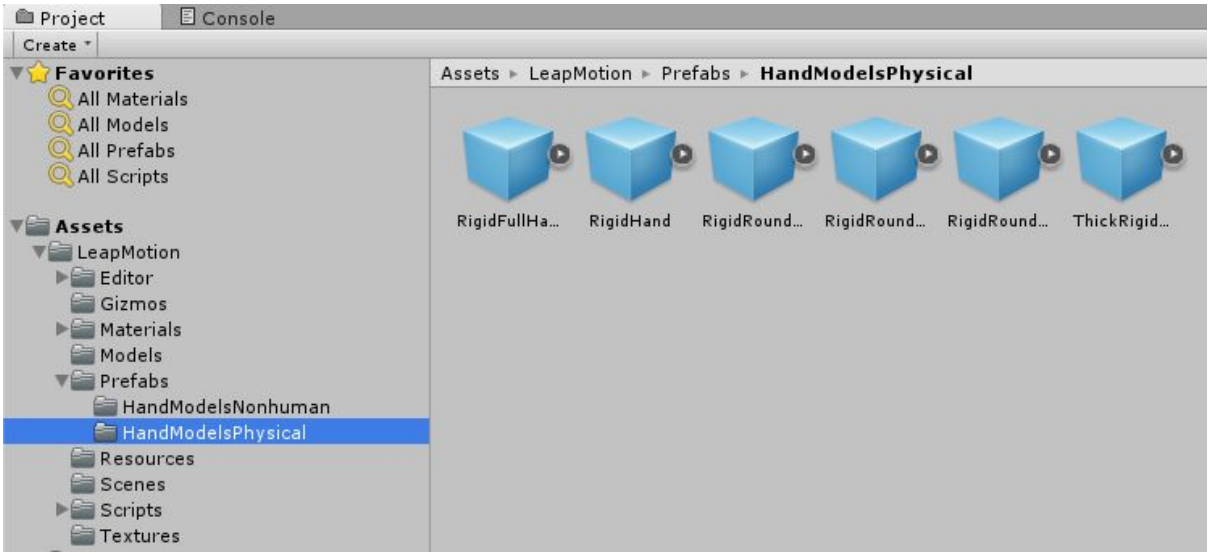
7) Once your physical representation is located, open VR Selector GUI. Set the *selector type* as Touch and drag the physical representation object to the *Selector object* field. Once you are ready, click on Add TouchSelector



8) You can add a Grab Trigger if you wish, just as you would normally.
9) Now your Leap Motion hand should be able to interact with VR elements via the bone(s) you chose! Note that the VR Display Button does not work with Touch Selectors yet -coming soon!- but the other VR elements should be usable.

Note: If you find that you are having to set this up every time for all your scenes where you want to use Leap Motion with VREasy, you can also use the Leap Motion HandModelPhysical prefab with the VRSelector GUI, so all the hands that use a particular physical model will come with a VR Touch Selector configured when you drag and drop them to your scene! To do so, instead of using the

LMHeadMountedRig prefab, drag the HandModelPhysical prefab you wish to use (LM comes with several of them) to your scene, and continue with steps 5 to 8.

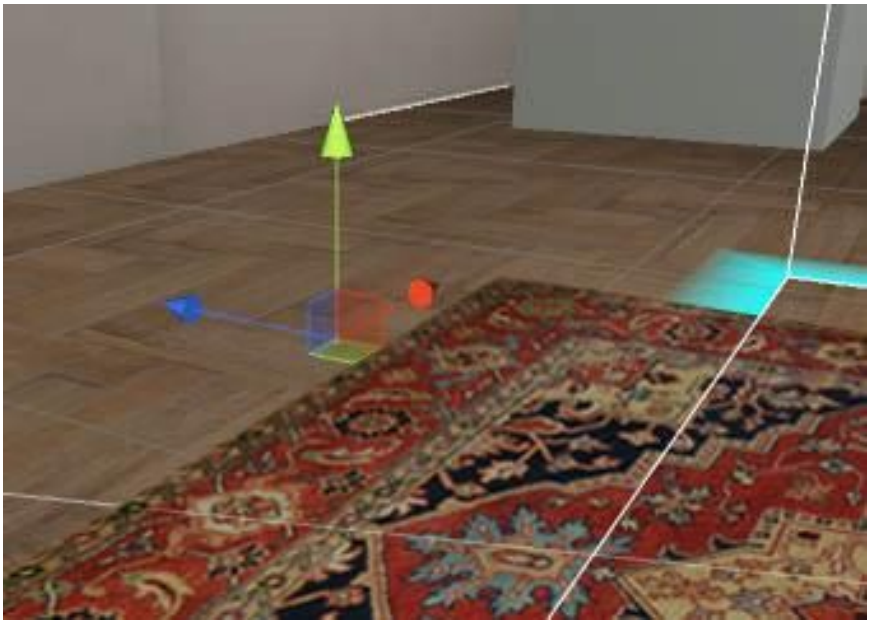


Note: See notice below for *Adding VREasy TouchSelector to other objects*. All TouchSelectors need their collider to be a trigger -current limitation. By default, Leap Motion physical hand models are not triggers. The VR Selector GUI will convert the object representation selected to a trigger collider.

OTHERS

Adding VREasy Sight selection to other objects (or HMD prefabs)

Theoretically one can add sight selection properties to any game object by dragging said object to the Selector object field in the VR Selector menu. Internally, the system works with raycasts along the Z axis, so wherever the SightSelector component is, it will *look for* VR Elements in that direction.



If one is to add SightSelector to the game object in the image, the ray would be casted along the blue arrow (Z axis, positive direction; also called forward vector).

This gives you a great deal of power, as you are not limited to selecting elements by looking at them; you can also select things by pointing at them, using this principles (PointerSelector coming up in future versions).

Adding VREasy touch selection to other objects (or controllers)

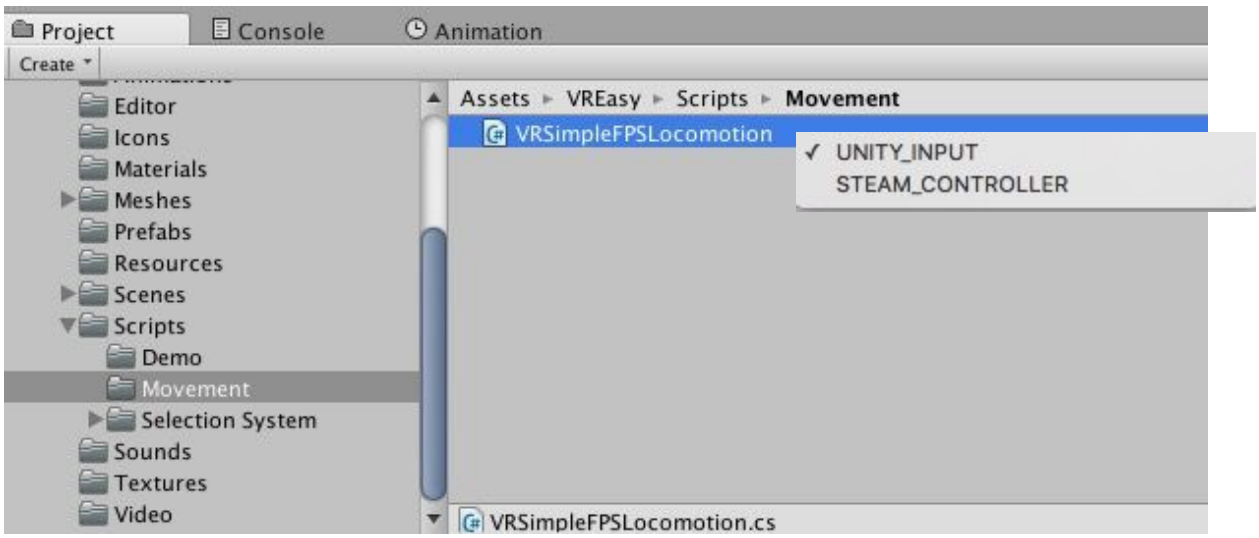
This is easily done via the VR Selector GUI. Just select whatever game object is being controlled by your Input as a *Selector object* in the VR Selector GUI -after having selected Touch as *selector type*.

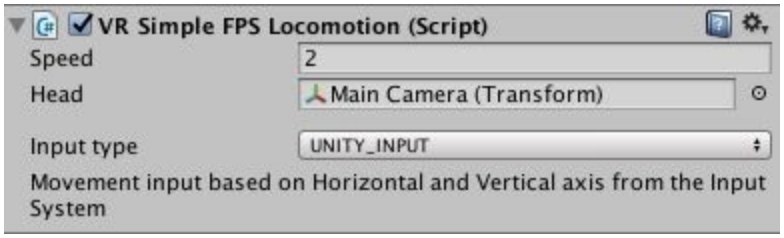
Note: the VR Selector GUI attaches a Rigidbody and a BoxCollider, which are required by the selection system to work. If your *Selector object* already has either of them (any collider would also work), it will use the settings in them; if not, it will add a kinematic rigidbody and a BoxCollider. Please make sure the size and position of the collider are adequate for the desired interaction.

IMPORTANT: At the moment the TouchSelector collider needs to be set as a trigger (automatically done when creating a TouchSelector via the VR Selector GUI. If this is a problem with your current set up (if you need your game object to still be able to collide physically with other objects) then attach a second collider to your game object and set that one to not be a trigger.

GUIDE FOR LOCOMOTION SETUP:

1. Attach the VRSimpleFPSLocomotion script to the parent Game Object of the HMD object in your scene. If using Unity support, this is the Main Camera; if using Steam prefab, this is the top parent, [CameraRig] or [SteamVR].





2. The VRSimpleFPSLocomotion requires a reference to the HMD object. This is the object that actually represents the head of the player, the HMD itself. For unity supported cameras, this is the Main Camera object itself. For Steam, this is the child object of the main prefab called Camera(head). *This step is required as we cannot directly move the HMD (this is anchored to the physical location of the player's head).* So drag the HMD object to the HEAD reference of the VRSimpleFPSLocomotion
3. Select the type of input you want. Unity input is for anything linked to the Input system as Horizontal and Vertical (keyboard arrow keys, gamepad controller directions...). If you want to use steam controller, you need one more step
4. (only for steam controllers) If you want to use a controller, just drag the controller ifself (SteamVR trackable object) to the tracked object field in VRSimpleFPSLocomotion; when using the steam prefab, this is either of the children objects called Controller(left) or Controller(right)