Currently tested working with Leap Motion Orion Core Assets 4.4.0 & Orion Hands Module v2.1.4 on Unity 2018.2

Developed by Ivan Bindoff

Contents

Installation

1. Open your Unity project.
2. If you have previously had an old version of the Leap Motion, ensure it is COMPLETELY REMOVED. This includes the various Leap motion DLLs etc. that may be hiding in a plugins folder.
3. Ensure that the Orion version of the Leap Motion tracking software is installed on your computer https://developer.leapmotion.com/orion
4. Ensure the official Leap Motion Orion Unity core assets package is added into your Unity project https://developer.leapmotion.com/unity
5. Ensure the official Leap Motion Orion Hands Module package is added into your Unity project https://developer.leapmotion.com/unity
6. Download and import this Avatar Hand Controller for Leap Motion from the asset store.
7. Done! The package will be installed into a LeapAvatarHands folder. You can check out a demo scene in the Scenes sub-folder.

Upgrading

Please note that, as with most Unity Asset Store assets, when you upgrade the Official Leap Motion and LeapAvatarHands assets you are recommended to completely remove the old version of the asset (by deleting the corresponding folder from your project) before installing the new version. This is because Unity assets are purely additive - we cannot release "patches" to the asset, so new files will be written and old files will be overwritten, but any existing files that should have been removed will remain.

Configuring your own avatar

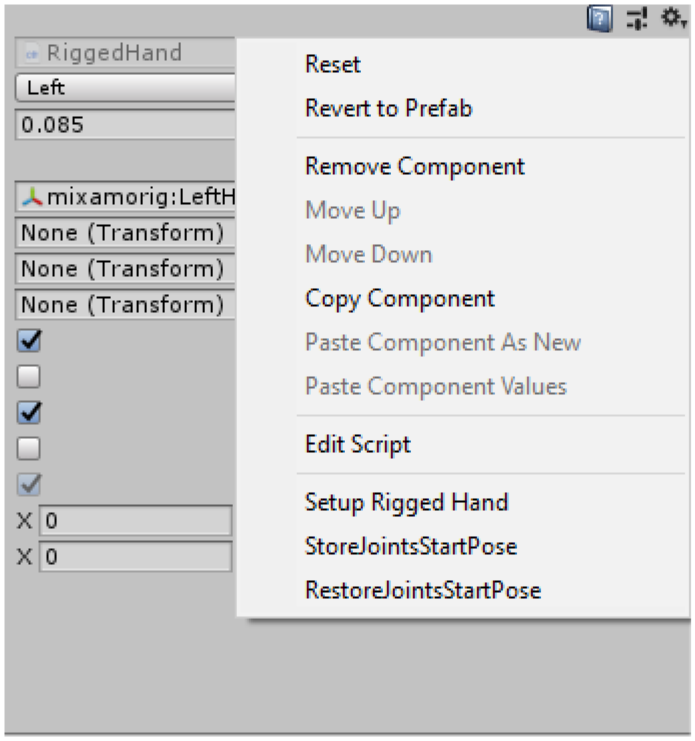
Adding this solution onto a player avatar does unfortunately require a fair bit of configuration.

For Desktop Mode (Leap Sensor placed on desk in front of you)

Ensure Virtual Reality Supported is disabled in the Player->XR settings of Unity.

1. Add your player avatar object to the scene - the player gameobject must be a properly rigged character with 5 fingered and 3+ jointed hands. I did all my testing with characters generated by Mixamo Fuse, and then rigged with the Mixamo auto-rigger.
2. Set the update mode on the Animator component of your character to “Animate Physics”. I suggest also using the Test Controller I provided, at least to start (alternatively you will need to at least make sure the IK pass is turned on, see Making your own Animator section below).
3. Add a child gameobject to your player gameobject (GameObject->Create Empty Child).
4. Position this child gameobject where you want the virtual Leap controller to sit, generally speaking you will want this to be about 20-40cm (about 1 ft) out from the avatar’s belly button.
5. Add the IKOrionLeapHandController script and the LeapServiceProvider script to this empty gameobject.
6. Go to the avatar’s left hand and add the RiggedHand script from the Leap Unity package to it.
7. Set the handedness to Left.

8. Click the cog icon top right of the component, then select the "Setup Rigged Hand" drop-down menu option, as shown in the image.



9. Repeat steps 5-7 for the right hand, replacing left for right.
10. Go back to the IKOrionLeapHandController gameobject and set the Avatar Left Hand and Avatar Right Hand fields to use the hands you just configured.
11. Go to the root node of the avatar and add the IKOrionActioner script, and tell it which gameobject you put the IKOrionLeapHandController onto.
12. Test your avatar by running the scene and placing your hands in front of the leap motion sensor.
13. You will often need to adjust the thumbs of your avatar to make them appear correctly - try setting the Model Palm Facing x parameter to -1 (for left hand) or 1 (for right hand) and z parameter to 0.
14. In some cases, other fingers may not have detected properly also. Experiment with the palm facing and finger pointing vectors to adjust to suit the avatar.

For VR Mode (Leap Sensor mounted to front of VR headset)

Setup the Project

Ensure that Virtual Reality Supported is turned on in the Player->XR Settings of Unity.

Setup the Camera

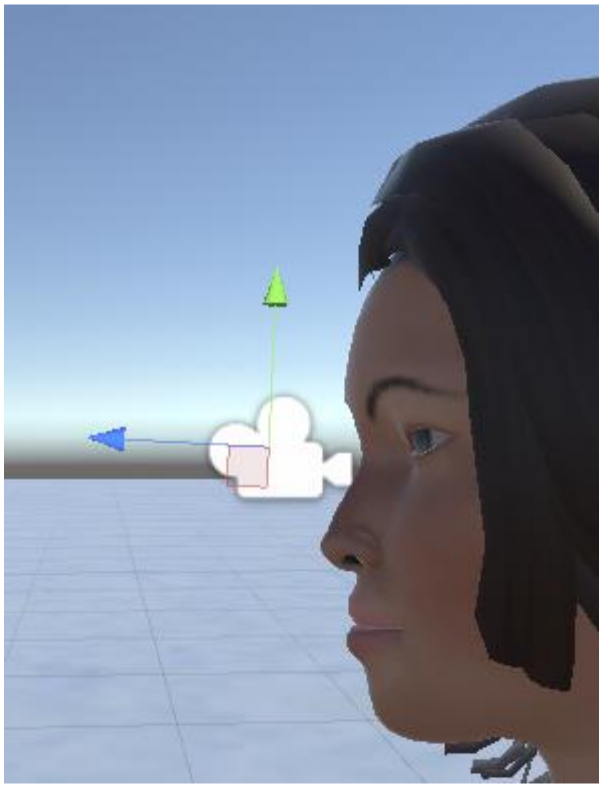
1. Make an empty gameobject called “Camera Parent”.
2. Move your Camera object onto the Camera Parent, the Camera should now be a child of Camera Parent.

3. Set the camera near clip plane to about 0.08. You can tweak this later if you are seeing the back of your avatar's eyes/nose/mouth when you're in VR mode.
4. Add the LeapXRServiceProvider to the camera.

Note that you will NOT need to use the XRHeightOffset script that is included in the Leap package, because the height of your camera will be determined by the avatar.

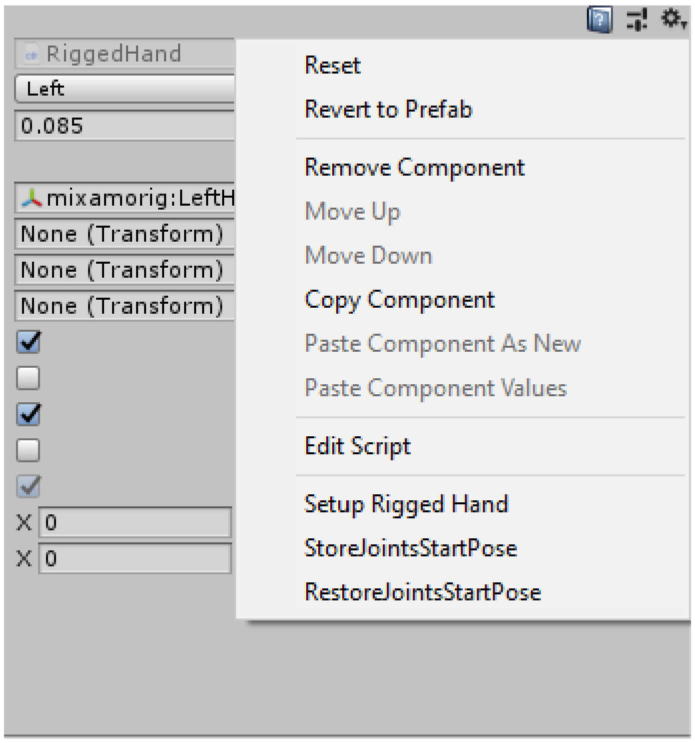
Setup your Avatar

1. Add your player avatar object to the scene - the player gameobject must be a properly rigged character with 5 fingered and 3+ jointed hands. I did all my testing with characters generated by Mixamo Fuse, and then rigged with the Mixamo auto-rigger.
2. Set the update mode on the Animator component of your character to "Animate Physics". I suggest also using the Test Controller I provided, at least to start (alternatively you will need to at least make sure the IK pass is turned on, see Making your own Animator section below).
3. Add the IKOrionLeapHandController script to the root node of your avatar.
4. Go to your avatar's head bone in the hierarchy and add an empty child gameobject (GameObject->Create Empty Child)
5. Call this gameobject "IdealMountPosition"
6. Position the IdealMountPosition a little forward of the avatar's eyes, as seen in the image below. You can adjust this position later if your camera is being positioned poorly relative to the avatar.



7. Go to the avatar's left hand and add the RiggedHand script from the Leap Unity package to it.

8. Set the handedness to Left.
9. Click the cog icon top right of the component, then select the "Setup Rigged Hand" drop-down menu option, as shown in the image. This may or may not work



depending on how your model has been rigged. If some of the fingers failed to autorig, you may need to add RiggedFingers to each finger, at the joint that represents the finger knuckle.
10. Repeat steps 5-7 for the right hand, replacing left for right.
11. Go back to the IKOrionLeapHandController gameobject and set the Avatar Left Hand and Avatar Right Hand fields to use the hands you just configured.
12. Test your avatar by running the scene and placing your hands in front of the leap motion sensor. Note at this point your hands will be stretched out and the character may be positioned poorly relative to the leap motion sensor. See Setup VR below to remedy this.
13. You will often need to adjust the thumbs of your avatar to make them appear correctly - try setting the Model Palm Facing x parameter to -1 (for left hand) or 1 (for right hand) and z parameter to 0. You may also need to tinker with the finger pointing. It’s a tedious process unfortunately.
14. In some cases, other fingers may not have detected properly also. Experiment with the palm facing and finger pointing vectors to adjust to suit the avatar.

Setup VR (including roomscale handling)

This is an experimental feature, and is intended as a starting point for you to reference when creating your own room scale VR experience with leap motion. It takes the camera (HMDs) position and rotation and uses it to adjust the position and rotation of the avatar so that the avatar will always be in the right place during roomscale experiences. It does not, however, deal with leaning/bending or crouching.

1. Add the RoomScaleVRAvatar script to your avatar's root gameobject.
2. Set the root transform, neck transform, and hip transform properties to reference the corresponding nodes on your avatar.
3. Set the target transform to point to your Camera/HMD object.
4. Set IdealMountingPosition to point to the IdealMountingPosition you created earlier, when setting up your avatar.

Physics hands

So that your hands can interact with physical objects in game, I have included a RigidIKRoundHand_L and RigidIKRoundHand_R prefab which will work with my IK solution.

For Desktop mode

Simply add these as children of the IKOrionLeapHandController gameobject and update the physics left hand and physics right hand references in the IKOrionLeapHandController script.

For VR mode

Place the RigidIKRoundHand_L and RigidIKRoundHand_R prefabs at the root transform (straight into the scene, not as a child node of another transform). Then update the left physics hand and right physics hand references in the IKOrionLeapHandController script to point to them.

NOTE: Physics hands can be tweaked by fine adjustments to the offset and wrist offset parameters in the RigidHands and RiggedHands components. The avatar hands are limited by the range of movement of the avatar's hands, whereas the physics hands are not. Consequently, it is sometimes difficult to get the physics hands and the real hands to sync up perfectly. The offset and palm width parameters of the RigidIKHands can be tweaked to improve accuracy, but it's still not perfect, particularly when your arms are at full stretch. You may need to create your own custom physics hands to match your avatar more perfectly.

Making your own Animator

It is important to note the two different layers I used in my example Animator, with IK Pass enabled on the base layer and blending set to override with a weight of 1.

Also, don't forget to set the Animator to Animate Physics! This tweaks the order of execution, allowing the Leap data to override the animation data.

If you make your own Animator and it's not working right, refer to the example to see how it's meant to be done.

Note when Using Hand Position Data at Runtime

Unity has a very particular order of execution. If you attempt to reference the hand transform in the Update method you will find that the data is incorrect, it will be based on where the currently playing character animation thinks the hand should be, rather than the leap motion hand data. The Animator IK pass happens after this.

As such, if you are trying to use the hand transform data at runtime, you should probably be doing that in LateUpdate(), after the Leap data and IK pass has occurred. Alternatively you can tweak the order of execution of the scripts to suit.

Gestures

I have included two example gestures in the VR scene. Teleporter and Shooter. Teleporter will teleport you to the position you’re gazing at when you extend the index finger of your left hand. Shooter will launch a capsule into the air when you extend all the fingers on your right hand.

When using the Teleporter script, place this on the Camera's parent transform. Set it's RayTransform (usually the camera), and set up your Ray Mask so it only works on Floor objects. For room scale you will also want to untick the Do Rotation box. You can see an example in the VR mode scene.