SteamVR Unity Plugin

Render Models
Button Input
Skeleton Input
Interaction System

For the most simple example of VR with tracked controllers see the sample scene at ***SteamVR/Simple Sample***. For a more extensive example including picking up, throwing objects, and animated hands see the Interaction System example at ***SteamVR/Interaction System/Samples/Interactions_Example***.

Render Models

SteamVR works with a wide array of headsets and controllers. When you have a tracked device in VR it's nice to have an accurate visualization of that device. Instead of shipping each individual model and texture for each controller with each application, SteamVR can manage this for you. We include a ***SteamVR_RenderModel*** component that you can put on a gameobject that will auto load whatever device SteamVR detects for a given Pose. Put a ***SteamVR_Behaviour_Pose*** on a gameobject to get the position and rotation of a controller and then a SteamVR_RenderModel to show the 3d model for that controller that will animate with button presses. See the simple sample scene or the ***SteamVR/Prefabs/CameraRig*** prefab for an example.

Input

Since SteamVR works with so many different input devices we've simplified the process of accommodating all these devices. Instead of referencing the individual buttons on one controller, with SteamVR Input you reference an action. This is part of Valve's implementation of the OpenXR standard. When you want the user to pick something up, instead of watching the trigger of a vive wand to be depressed to a value of 0.7 or the grip of an Oculus Touch controller to be depressed to a value of 0.6 you just wait until your "Grab" action gets set to true. As a developer you can setup default buttons and thresholds for your Grab action in SteamVR. Then when users get to run your application they can change those values to suit their preferences. This also helps you support future input devices that haven't been released without requiring you to write additional code or submit updates.

As an example, this is how it worked before SteamVR Input:

```
private const float oculusDepressValue = 0.3f;
private const float viveDepressValue = 0.7f;
private const float knucklesDepressValue = 0.1f;
private SteamVR_Controller.Device controller;
0 references
private bool CheckGrab()
{
    string model = UnityEngine.XR.XRDevice.model != null ? UnityEngine.XR.XRDevice.model : "";
    if (model.IndexOf("Rift") >= 0)
    {
        return (controller.GetAxis(Valve.VR.EVRButtonId.k_EButton_Grip).x > oculusDepressValue);
    }
    else if (model.IndexOf("Vive") >= 0)
    {
        return (controller.GetAxis(Valve.VR.EVRButtonId.k_EButton_SteamVR_Trigger).x > viveDepressValue);
    }
    else if (model.IndexOf("knuckles") >= 0)
    {
        return (controller.GetAxis(Valve.VR.EVRButtonId.k_EButton_Grip).x > knucklesDepressValue);
    }
    else
    {
        throw new System.Exception("New controller?!?!");
    }
}
```

This is how it works now:

```
public SteamVR_Input_Sources handType;
public SteamVR_Action_Boolean grabAction;
0 references
private bool CheckGrab()
{
    return grabAction.GetState(handType);
}
```

For a more in depth tutorial on this system check out the SteamVR Input guide included with this plugin (SteamVR Input.pdf) or this tutorial online: https://steamcommunity.com/sharedfiles/filedetails/?id=1416820276

Skeleton Input

With the Knuckles EV2 release Valve has released a system to get skeletal hand data independent from the controller you're using. Some controllers have a high fidelity for hand tracking data with individual joints, some controllers only have buttons that we use to approximate joint data. We use whatever is available and give you two major sets of data:

With Controller: Our best approximation of where the joints in your hand are wrapped around your controller in the real world.

Without Controller: Based on the With Controller data we estimate how open or closed your hand is as if you were trying to hold your hand flat or make a fist.

Included in this plugin are example models of gloves that you can freely use in your products. In addition we have a helper component that takes the data from SteamVR and applies it to these gloves: ***SteamVR_Behaviour_Skeleton***. For an example of this in action see the Interaction System sample scene.

Interaction System

After Valve released The Lab we took the learnings from that project and created an Interaction System that others could use in their own projects. This system has been updated since then to use SteamVR Input and the new SteamVR Skeleton Input systems. This system can serve as an example of how to use the these new systems. It includes the following examples:

Interaction with Unity UI elements
Pickup, Drop, and Throw
Multiple variations on throwing velocities
Bow and Arrow
Wheel interactions
Simple buttons
Variety of Skeleton Input examples
Teleporting
Using held objects
SteamVR Input