IT 4001: Virtual Reality

Assignment 10: Grabbing Objects with Oculus Touch

If you use your last assignment (Ball Game with Oculus Touch) as a starting point, you've already had the following SDKs imported, and you can skip step1 and go directly to step2.

Step1: Download, unzip and import the following SDKs.

You need to accept Oculus Terms and Conditions, then clicking download. Once you have them downloaded, you can go ahead and extract the archive.

- Oculus Utilities for Unity https://developer.oculus.com/downloads/package/oculus-utilities-for-unity-5/
  This will give you a new folder called "OVR" in your Assets folder. Within this folder, locate Prefabs/OVRCameraRig and drag it into the scene. If you had previously used our MeMyselfEye prefab in the scene, make sure it's deleted or disabled, as OVRCameraRig is replacing it.

- Oculus Avatar SDK https://developer.oculus.com/downloads/package/oculus-avatar-sdk/
  Locate a file called *OvrAvatar.unityPackage* in Unity folder. Take that file and drag it into your Unity project to import its assets. This will give you a new folder called "OvrAvatar" in your Assets folder. Within this folder, there should be a "Content" folder, expand that and find the "Prefabs" folder within it. You should see two assets in it: LocalAvatar and RemoteAvatar. We want LocalAvatar, as this is what we will use to display the hands.

- Oculus Platform SDK https://developer.oculus.com/downloads/package/oculus-platform-sdk/

When you drag the LocalAvatar into your project, make sure it is as its game object, not under the OVRCameraRig in your scene. You can place LocalAvatar and OVRCameraRig both at (0, 0, 0) position. Make sure your Touch controllers are on, go ahead and run your project. You should immediately be able to see your hands in your scene. This should include the ability to use your fingers and see them in virtual reality. The Avatar SDK provides quality hands or controllers for your app, with no need to create any assets yourself.

Note that you will see an error saying you must supply an Oculus Rift App ID for your project. You can ignore this for testing, but will need to add one before you can create a final build. If the floor is set at origin, it might occlude the hands. Simply lower the floor slightly (i.e. set Y to -0.5).

Please note that the Avatar SDK only covers visual representation of the hands. Even though it's really exciting to see your hands in VR, you won't be able to interact with anything quite yet, as your hands do not have any collisions associated with them.

Step2: Grab and throw objects.

Add Sphere Colliders to the LeftHandAnchor and RightHandAnchor objects under the OVRCameraRig, and set their radius to 0.075 as this seems to be a good size for them with respect to the hand models. Then check the "Is Trigger" checkbox on them as well. This will ensure that when the hands collide with something that they are able to grab them. Finally, add a Rigidbody to each hand, deselecting the "Use Gravity" checkbox and selecting the "Is Kinematic" checkbox, that way the player can move the hands around themselves.

Create a new script called Hand.cs. Paste the following into your Hand.cs' class:

```
public class hands : MonoBehaviour {

    public enum State
    {
        EMPTY,
        TOUCHING,
        HOLDING
    };

    public OVRInput.Controller Controller = OVRInput.Controller.LTouch;
    public State mHandState = State.EMPTY;
    public Rigidbody AttachPoint = null;
    public bool IgnoreContactPoint = false;
    private Rigidbody mHeldObject;
    private FixedJoint mTempJoint;

    // Use this for initialization
    void Start () {
        if (AttachPoint == null)
        {
            AttachPoint = GetComponent<Rigidbody>();
        }
    }

    void OnTriggerEnter(Collider collider)
    {
        if (mHandState == State.EMPTY)
        {
            GameObject temp = collider.gameObject;
            if (temp != null && temp.layer == LayerMask.NameToLayer("grabbable") &&
temp.GetComponent<Rigidbody>() != null)
            {
                mHeldObject = temp.GetComponent<Rigidbody>();
                mHandState = State.TOUCHING;
               // print("touch");
            }
        }
    }

    void OnTriggerExit(Collider collider)
    {
        if (mHandState != State.HOLDING)
        {
            if (collider.gameObject.layer == LayerMask.NameToLayer("grabbable"))
            {
                mHeldObject = null;
                mHandState = State.EMPTY;
            }
        }
    }
```

```
 // Update is called once per frame
    void Update () {

        switch (mHandState)
        {
            case State.TOUCHING:
                if (mTempJoint == null && OVRInput.Get(OVRInput.Axis1D.PrimaryHandTrigger,
Controller) >= 0.5f)
                {
                    mHeldObject.velocity = Vector3.zero;
                    mTempJoint = mHeldObject.gameObject.AddComponent<FixedJoint>();
                    mTempJoint.connectedBody = AttachPoint;
                    mHandState = State.HOLDING;
                }
                break;
            case State.HOLDING:
                if (mTempJoint != null && OVRInput.Get(OVRInput.Axis1D.PrimaryHandTrigger,
Controller) < 0.5f)
                {
                    Object.DestroyImmediate(mTempJoint);
                    mTempJoint = null;
                    throwObject();
                    mHandState = State.EMPTY;
                }
                break;
        }


    }

    private void throwObject()
    {
        mHeldObject.velocity = OVRInput.GetLocalControllerVelocity(Controller);
        mHeldObject.angularVelocity =
OVRInput.GetLocalControllerAngularVelocity(Controller) * Mathf.Deg2Rad;
        mHeldObject.maxAngularVelocity = mHeldObject.angularVelocity.magnitude;
    }
}
```

Some reference information about Unity FixedJoint:

Fixed Joints restricts an object's movement to be dependent upon another object. This is somewhat similar to Parenting but is implemented through physics rather than Transform hierarchy. The best scenarios for using them are when you have objects that you want to easily break apart from each other, or connect two object's movement without parenting.

Tutorial Physics Joints: https://unity3d.com/learn/tutorials/topics/physics/physics-joints

Attach the Hand.cs script to each Hand anchor – LeftHandAnchor and RightHandAnchor under OVRCameraRig. Ensure that the Controller dropdown is properly set for each hand ("L Touch" for Left hand and "R Touch" for Right hand).

Next, add some objects to the scene - i.e. a Cube and add a Rigidbody to the cube. Ensure that it is on a layer called "grabbable". You should know by now how to add a layer, and set an object to be a certain layer.

Turn in your completed project (zipped folder) on Canvas, and demo to the TA/Instructor.