Team 8 Group Submission

John Cole, Sean Barton, Josh Spain, Ben Knepper

Setting up the OSVR and Hydra Controllers with Unity

First off is obviously hooking everything up. The OSVR cables are pretty self-explanatory, as each connector has a distinct shape. Everything connects to the small square. There is also a cord with MicroUSB and USB ends that is for the IR tracking camera, which I don’t think we’ll use, as well as a cord that attaches to the power cord for the IR camera power.

At this point, I believe the devices are pretty much plug-and-play. Everything should just work once its plugged in, and you won’t have to do anything special once you’re in Unity. The OSVR just shows what’s on the screen, though, so you need to make the game full-screen when you’re testing it with VR. I’m not sure how one would go about playing the game in true full screen on the OSVR. As for how to get to this point, I had to install drivers for everything as well as import plugins for Unity. I’m also not sure if the Hydras took extra setup.

To make a scene VR compatible, you just need to drag-and-drop the VRPlayer prefab into the scene. It includes the player entity, OSVR camera, and Hydra-connected hands. This prefab was made from assets from the OSVR and Sixense (Hydra) packages. Specifically, it includes the VRFirstPersonController, ClientKit, and Recenter prefabs from the OSVRUnity->Prefabs folder and the SixenseInput script from the SixenseInput folder. There is also a HandsController object which was a prefab created from an object of the same name in the SixenseInput demo scenes. In order to make the hands follow the camera, they are a child of VRPlayer->VRFirstPersonController->VRDisplayTracked->VRViewer0 object.

Finally, to work with input from the Hydras, you need to wire into inputs in the Sixense package, since it apparently doesn’t use the normal Unity input. A summary of the basic input functions can be found here: <http://blog.dsky.co/2015/05/16/razer-hydra-input-in-unity3d-sixense-input-control-syntax/>. Note that the “i” in SixenseInput.Controllers[i] refers to the left or right controller; 0 is left and 1 is right. The results of these functions can be combined with the normal Unity input in order to support both using the Hydras and the keyboard/mouse. For example, you can add the floats from SixenseInput.Controllers[0].JoystickX and Input.GetAxis(“Horizontal”) or “or” the bools from SixenseInput.Controllers[1].GetButton(SixenseButtons.ONE) and Input.GetButton(“Jump”).

One thing we struggled with briefly after the initial setup was getting the hydra controller’s built in “hands” objects to follow the camera around properly. After some debugging we were able to child them to the camera (which is a child of our player object) which made the hands behave accordingly. Finally, we were able to make prefabs of all of the special hydra and OSVR utility files so that hopefully, when we develop new scenes it will be a seamless experience.