

Videoplayer, Pens and Other Prefabs

<https://www.youtube.com/embed/l2t8V55PLoI>

I figured I'd dedicate a separate short lesson to VRChat specific prefabs, because there are a few that I know a lot of people want in their worlds.

Many VRChat prefabs can be found on the VRCPrefabs database. Note that SDK2 prefabs will not work in our SDK3 world, only Udon Prefabs will work.

Many prefabs require UdonSharp to work; this compiles C# code to Udon. You don't need to know how it works, just make sure you import it before importing an Udon prefab that requires it.

One of the most popular features of small worlds is a video player. The most popular of these is Merlin's UdonSharp player (which we will be using), but some other ones are VRCUdon's and there is also ProTV which is quest compatible. First, we need to import UdonSharp. Grab the unitypackage from the Github page under releases for both UdonSharp and UsharpVideo. There should be a videoplayer prefab in the UsharpVideo folder in the project, drag that into the scene, unpack and you're done!

You can change some options on the player, like whether control is locked to the master of the world or whether anyone can paste video links. You can change the UI style by going to the ControlsUI gameobject in the prefab, clicking on the style file and selecting a different one in the folder or you can make your own.

Prefabs with Udon elements can cause conflicts, so it's best to unpack your prefabs in your scene to ensure they work properly. You can right click on the prefab in the hierarchy and click unpack prefab; this will make your object no longer a prefab.

Another popular prefab is Qvpens. Grab these from their page on booth and make sure it is the Udon version. After importing, drag the prefab into the scene.

If you want a low quality mirror without the skybox, you can use [VRCPlayersOnlyMirror](#). It comes with a UI pre-setup.

The standard shader isn't very good but thankfully there are community created replacements that are much more intuitive and fully featured. Some of these include z3y's shaders or Silent's Filimented, but the one we are going to use is Mochie Standard because it is the easiest to install and tells you if your grayscale textures are set to sRGB and provides a button to automatically fix it if they are. It uses the industry standard roughness maps instead of smoothness maps, and gives greater control over the parallax mapping for heightmaps, among many other features.

Water shaders are also something many world creators want, and Mochie's shader package also comes with a decent water shader. Other VR specific water shaders I would recommend are Red_Sim's water shaders, Silent's Clear water, and Norbien's water, among others. Note that to have refraction, the shader must use a grabpass which is computationally expensive and does not work on Quest. Many water shaders also require a depth pass, which necessitates that a realtime light with shadows be in the scene. There is a DepthPass prefab included in VRWorldToolkit that is set to have the least impact on performance.

As for my personal opinions on them, I think Red_Sim's have the best caustics (Mochie's water caustics look too much like a voronoi texture) and Norbien's seem to have the best performance and also works with Quest and doesn't require a real time light for a depth pass.

<https://github.com/zulubo/SpecularProbes> can make decent looking baked specular highlights. Has limitations (like if the specular highlight is supposed to be occluded, doesn't look as good as realtime specular highlights) but there are definitely certain scenes and aesthetics that stand to benefit from it.

For those who are interested, if you want to convert Unity C# scripts to UdonSharp, it typically isn't very difficult. Typically you can change the namespace from `MonoBehavior` to `UdonSharpBehavior`, plug it into an udon component (UdonSharp will automatically fill in the rest) and it should work. You can check the UdonSharp class exposure tree by going to `Window>UdonSharp>Class Exposure Tree`. If you run into problems or have more questions visit the Discord server linked in UdonSharp's Github page.

If you plan on writing Udonsharp scripts I would also recommend UdonToolkit, which allows you to create custom inspectors for your scripts (like I did on my fork of Nova_Max's Daynight cycle prefab!). It helps make them easier to use for not only yourself but also anyone else if you plan on making it available.

Some prefabs don't come in a unitypackage but in a zip file with a package.json file inside. To import these, open the package manager, click the plus icon, then click the package.json file and open it.

Revision #2

Created 2022-02-19 23:03:52 UTC by Iconoclass

Updated 2022-05-23 23:30:02 UTC by Iconoclass