

External Assets and Prefabs

<https://www.youtube.com/embed/r9l0jaRbL4o>

Again I have to backtrack at the first part of the lesson because it turns out a topic I chose not to cover last episode is actually of importance because our scene's lighting comes primarily from the environment (indirectly) rather than from a light source component. If you look at some parts of the wall you may see that some of them look a little blotchy. This is being caused by both the indirect and environment samples and by the filtering in the lighting tab.

Rather than simply explain it I feel it is best to show you to truly understand how the technology works. Slight trigger warning, I'm going to disable the filtering and the scene is going to look very noisy and ugly. For most of you it will be unpleasant to look at but if it disturbs you in any way you can skip ahead ten seconds. This is what the lighting looks like with the filtering turned off.

The reason it looks so noisy is because of how ray tracing works; essentially a higher sample count means less noise. However, under filtering>advanced, you can see that there are denoising options. These reduce noise by using AI denoising algorithms. By increasing the number of samples, the baked result has less noise, and there is less blotchiness in the bake, at the expense of render time. From what I could read and from my testing, it appears that the environment samples is bounded by the indirect samples, so I'm going to increase both by about fourfold to 2000. If you are in a state where you are baking a lot just to preview your lighting you will probably want to decrease this until your final bake.

In scenes where the primary light sources are from light components, this typically isn't something to worry about. It's only because our primary light source is the environment that this becomes something worth considering. Direct lighting will always be far more accurate, far less noisy and take far less time to bake than indirect lighting.

Onto the main topic of this lesson. For those who don't know, an asset is just anything that goes into a 3D scene. There are websites you can download 3D assets from (always make sure you are legally allowed to use them) but the place I would typically check first is the Unity asset store. You can either browse it from inside Unity (windows>Asset Store), but the in-editor browser is slow, so I recommend using a dedicated browser. Go to the following link: <https://assetstore.unity.com/>

The convenient thing about the Asset store is that the assets are made specifically for Unity. I say this for two big reasons: one, many of the assets are simply drag and drop and don't require much additional setup or configuration (if any). Two, many 3D models that you find on other websites will have polygon counts that would be absurd for VR games (>20,000 polygons for simple objects).

This is because they are not intended for games but for movies or 3D renders which don't share the same technical limitations. The recommended maximum polygon count for Quest worlds is 50,000 triangles according to the official VRChat documentation, and even if you don't plan on uploading to Quest I would still keep a soft polygon limit in mind. Saving a few dozen here or there would be negligible for performance, but it wouldn't make sense to have a single object with 50,000. If you still want to use that object and it doesn't come in a low-poly version I would recommend taking it into Blender and either using the decimate modifier (which is quick and dirty) or baking to lowpoly.

The first asset we should search for is a proper skybox. Just type skybox into the search bar, check free assets only, and I'm going to use this allsky free pack for this tutorial because it is free, popular, and high quality. After it is in your asset library, click open in Unity which will take you to the package manager. You can also open it by going to Window>Package Manager and then making sure the dropdown at the top left next to the + dropdown is set to "My Assets". Now click on download and import and they will be in your project. Find the overcast skybox material and drag it onto the skybox in the scene. It looks a little dark so I'm going to increase the exposure to the point that the highlights are not overexposed.

Our room looks kind of barren, why don't we search for some furniture? Pick out something you like. I'll search for a sofa. I'll be using this sofa pack for this tutorial but even if you're not you can still follow along with your model of choice. Since this is your room I encourage you to make it your own, and that means using modelling, textures, lighting, and assets that fit your scene. That means you shouldn't put in elements that would clash with each other, and every part should ideally complement every other part.

First, add the model on the page. Download and then import through the package manager. Once imported look for a prefabs folder, click on the one you want, duplicate it, and drag it in the scene.

Before we get into that it's necessary to know what a prefab is. A prefab is a gameobject with all its components intact. The advantages of working with prefabs is that if you have a lot of the same gameobjects in a project and you want to edit them all at once, you can do so without having to individually edit each one.

Say we want all our windows to be the same. Instead of having to edit each one individually, we can make them a prefab by dragging our parent window object from the Hierarchy into the project tab. You should now see that the object icon in the Hierarchy turns from a hollow gray to a solid blue! You can edit prefabs by double clicking on them in the project tab, or by right clicking the prefab in the hierarchy and clicking open prefab asset, and the scene view will focus on just the prefab. Edits you make from here will apply to all instances of the prefab. Exit out of this view with the arrow at the top left of the scene tab. If you make a change in one of the instances of the prefab you can right click on that parameter in the inspector and click "apply to prefab". If you want to make one of the instances of the prefab no longer a prefab, you can right click on it in the hierarchy and click unpack prefab. You can also put prefabs inside of prefabs; unpack prefab only unpacks the parent prefab while unpack completely also unpacks any nested prefabs.

The sofa prefab I got didn't come with any colliders. Let's add some by double clicking on the prefab in the project tab, going to the gameobject that has the mesh renderer component (so the colliders will conform to the bounds of the mesh when they are first added) and adding box colliders and shape them as a simple boxed version of our mesh. For the colliders for the cushions of the sofa I'm going to make one, click the 3 dots, copy component, and then paste component as new. This way the height of the two colliders will be exactly the same and players won't notice when walking from one collider to the other.

For complex objects, you could make a simplified collision mesh with probuilder (or blender), make it a child of the model gameobject, and disable its mesh renderer and add a mesh collider component. The primitive collider components are more optimized and reliable so it's preferable to use a compound of those when possible.

Our sofa is also static, so don't forget to tick that box.

Let's add some seats to the sofa. Search for VRCChair, duplicate VRCChair3 and then double click on the newly created VRCChair4. Delete the gameobject with the chair mesh and scale down the collider vertically. Open the sofa prefab, drag in the new VRCChair and place it under where you want there to be a seat. Make it face the correct direction; make sure that the tool handle is set to local rotation, then the blue arrow, the z axis, is forward. Ensure that the top part of the collider is above the collider of the sofa. The seat and exit gameobjects are empty and are only used for the transform where the player will sit and exit; you can adjust them if you like. Duplicate the chair prefab if you want more seats.

There is a VRChat layer called walkthrough, and it allows players to walk through objects that have colliders, while physics objects like pickupables still react to them. Since elevation change can be one of the most disorienting parts of VR, especially for new users, I tend to set many smaller static objects to this layer by default. In a later tutorial, I'll show how to toggle these objects between the Walkthrough and Default layers.

Also, find the model file for the object in the project view (you can get to this by clicking on the mesh in the mesh renderer component of the object). In the inspector under the model tab, ensure that generate lightmap UVs is checked. You will have to do this for every model you import that is set to be lightmapped (unless it specifically states that it comes with UV2s for lightmapping).

Imported textures are not compressed! Don't forget to compress them!

Don't forget to move your light probes so that they are not in places which dynamic objects can't reach or places where they will be blocked from receiving any light (like inside of objects)!

As for models from other sites, they require more setup than ones from the asset store. Typically there will be a model file and separate texture image files, so you will have to create a new material and plug in the textures yourself. The ideal 3D model format to import into Unity is .fbx, but .obj also works for static meshes. If you are downloading from an external site, make sure it is one of these formats. DO NOT import .blend files into Unity and drag them into your scene, there will likely be issues. And don't forget to check Generate lightmap UVs if the object is going to be

lightmapped!!!

You can use a 360 degree image or an HDRI as your skybox. I'm going to go to a website called polyhaven which has public domain HDRIs. When you find one that you like, download the 2K EXR (both are compatible with Unity but EXR is preferred). Drag into the project window, and it looks like a 2D texture. We want to project this texture like a sphere, hmm, when have we done this before? It sounds kind of similar to the cubemap of the reflection probe. To change the projection of the texture, click on it and in the inspector change the texture type from 2D to Cube. Now you can just drag the texture onto the sky and Unity will create the skybox material for you. You could even combine a higher intensity skybox with a low intensity directional light for artistic effect.

Note that the projection and scale of 360 degree pictures will not look correct as a skybox, and that there will not be any parallax when the player moves. This makes rooftop shots the best 360 images to use since when viewed through a window you naturally wouldn't see much parallax.

If you want world music or sounds in your world, drag an audio file into your project and then drag it into the scene, and it will automatically come with an audio component. In the inspector click Add Component and search for VRCSpatialAudioSource (which makes the sound work in VRChat). If your sound is world music, you probably don't want the sound to be coming from a certain point in the world, so go under advanced options and uncheck Enable spacialization.

For better compression, click on the audio file, set the format to Vorbis and drag down the compression slider. If the compression is noticeable you can drag it up until it isn't.

If your sound is spacialized then having a stereo audio file won't matter so set it to mono for a reduced file size.

And that's about it for this episode. This marks the halfway point in the series; the tutorials up until this point were mostly just generally learning the Unity software with an emphasis on interiors. From here on out, the tutorials will mostly be about learning implementations of Unity components specifically for VRChat. In the next lesson, we will go over how to add a mirror and a toggle for that mirror using world space UI.

Revision #1

Created 2022-02-19 22:49:22 UTC by Iconoclass

Updated 2022-02-19 22:50:32 UTC by Iconoclass