1 LIV SDK Documentation for Integration with Unity

1.1 Introduction

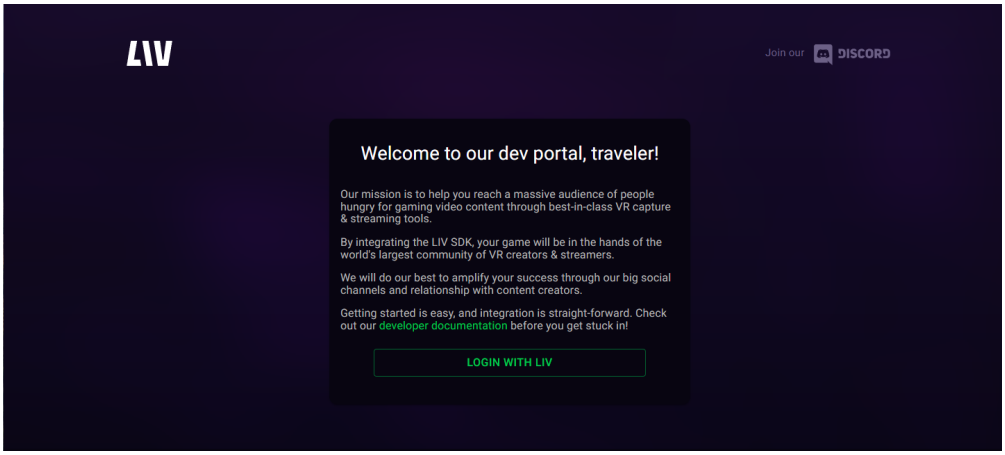
Here is how I integrated LIV SDK into Unity for Mixed Reality Capture. First of all, check the official documentation for LIV for the most detail. This is just the process that worked for me.

- LIV SDK Official Documentation
- LIV SDK Support for Unity YouTube Tutorial

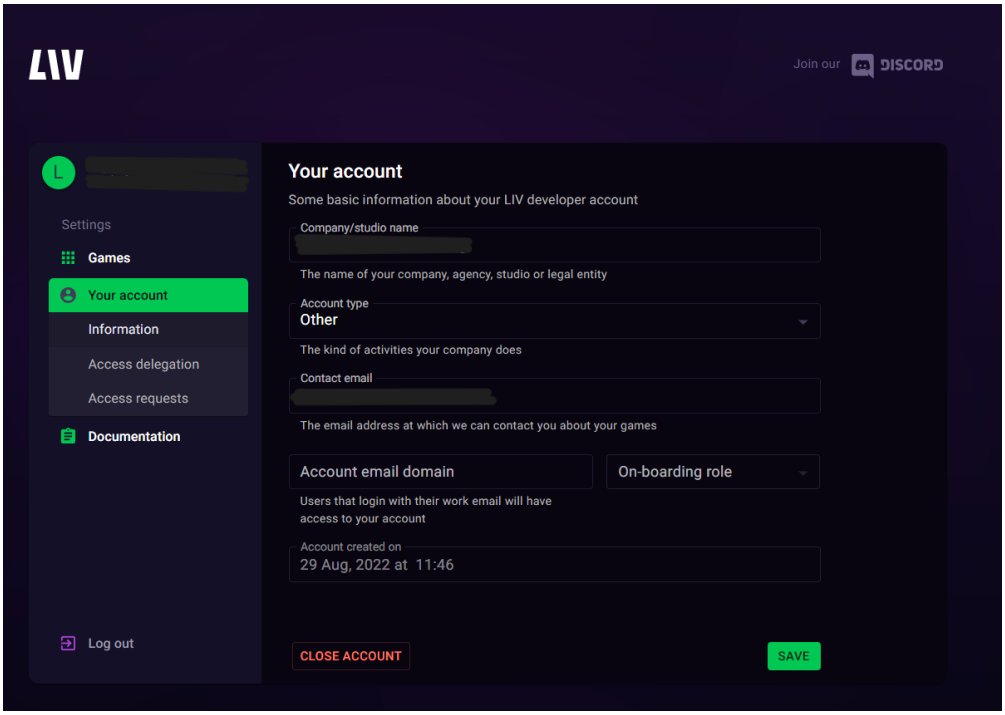
1.2 Downloading the LIV SDK

The LIV SDKs can be downloaded through the LIV Developer Portal.

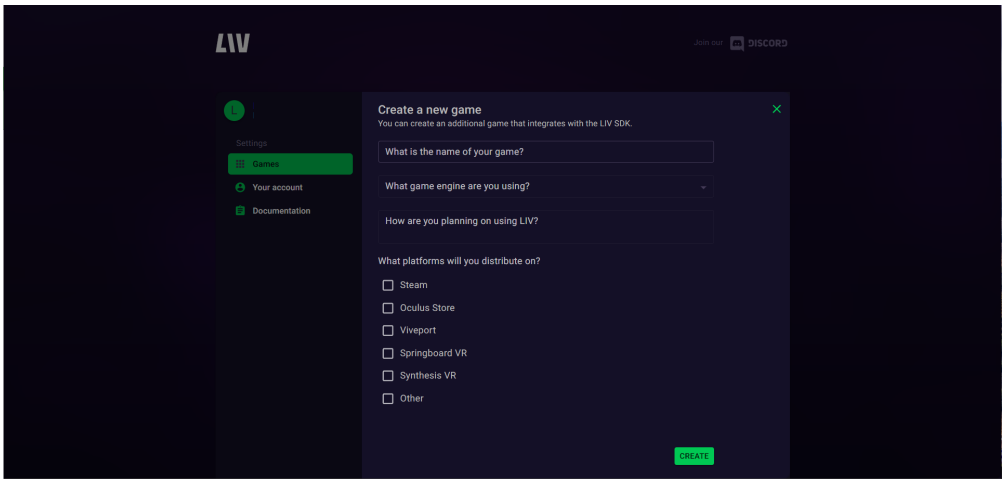
1. Register at dev.liv.tv with your work email.



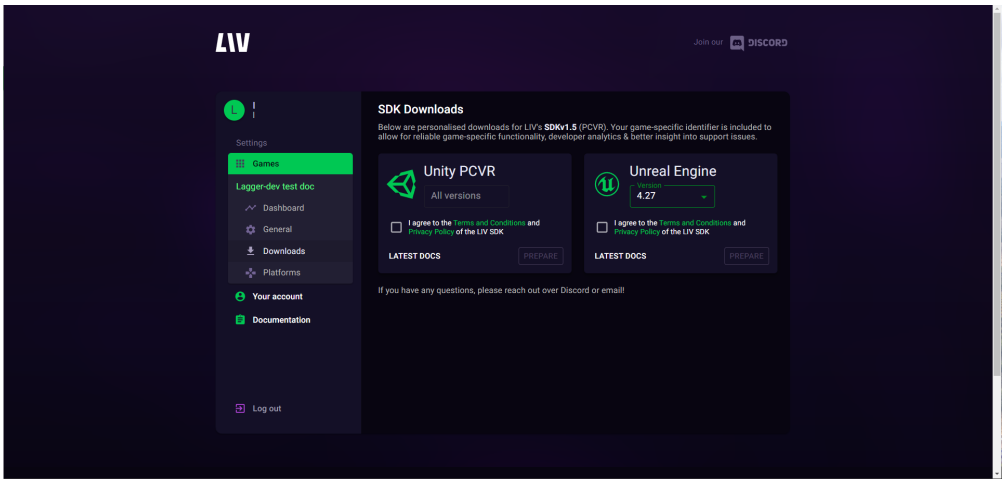
2. Create your account, using your business contact.



3. Create your game record with LIV.

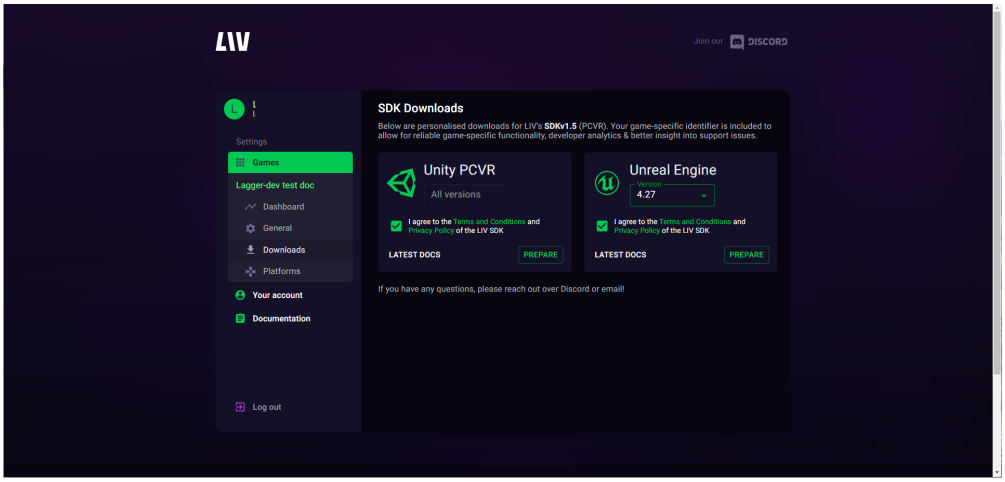


4. Select your game and navigate to Downloads.

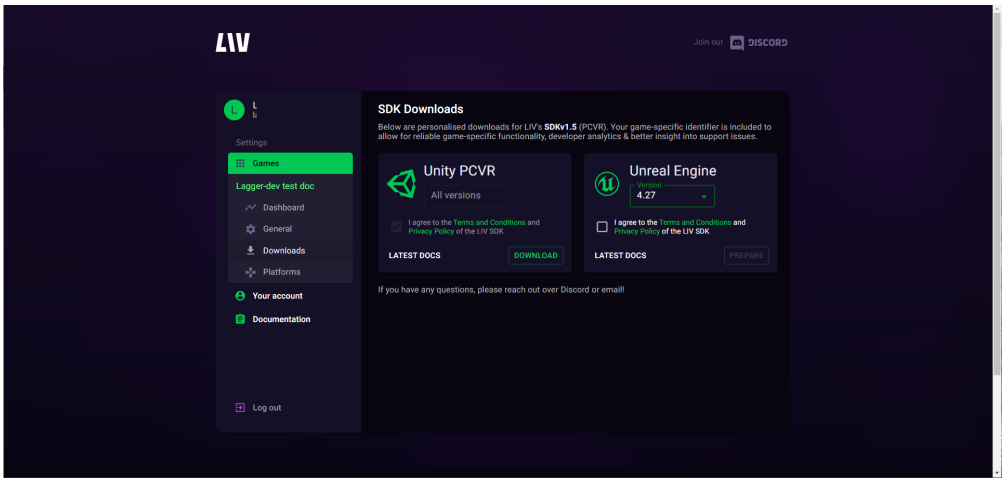


5. Agree to the Terms of Service & Privacy Policy.

6. Generate your app specific SDK by selecting **Prepare**.
   Note: If you create a new game you'll need a new SDK. Select **Prepare**.



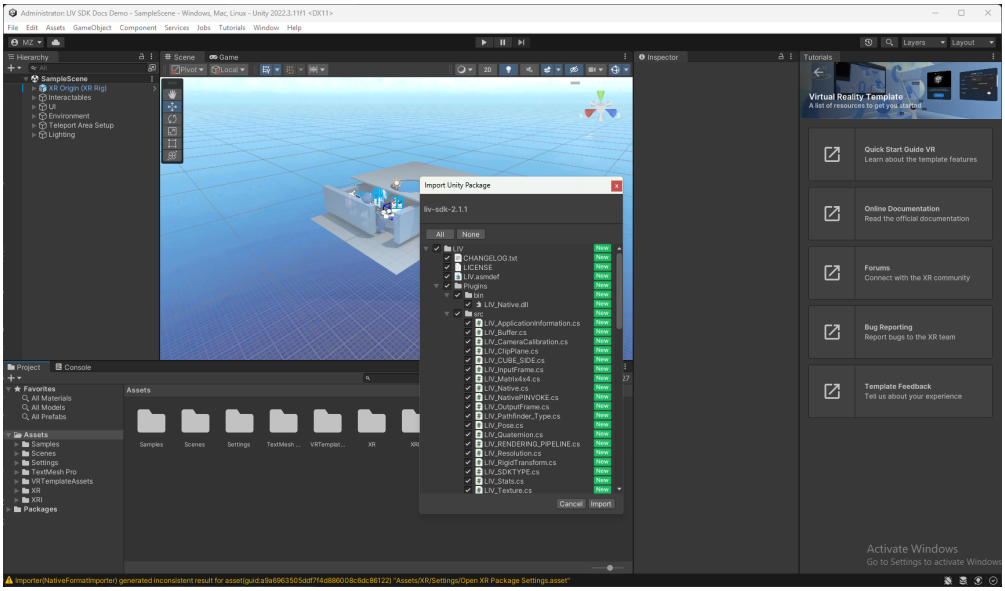
7. Download the SDK and you're ready to move onto installation.



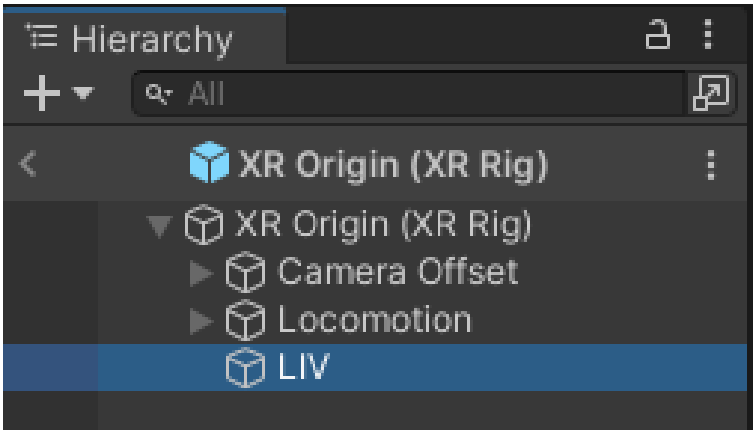
1.3 Unity SDK Installation and Configuration

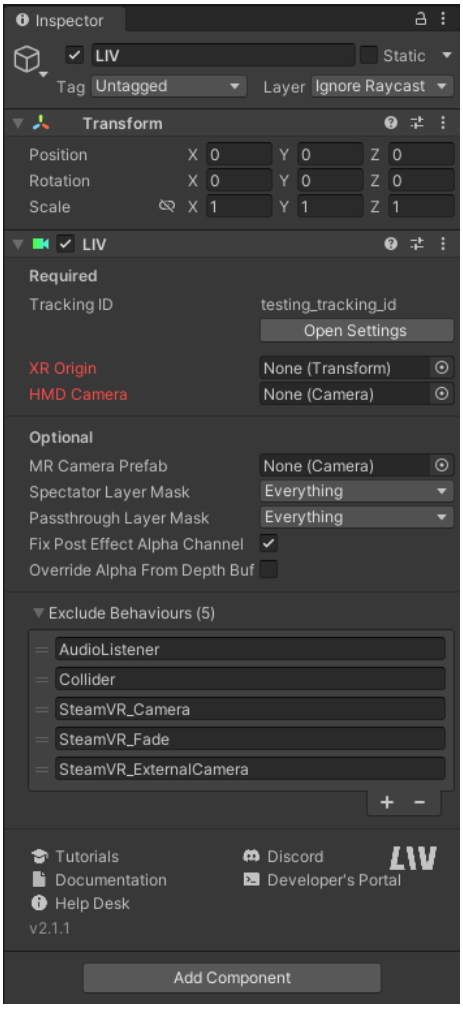
I'll add the LIV SDK to a **VR Core** template project in Unity for demonstration. The Unity Editor version I'm using is **2022.3.11f1**. The setup should be similar to this for other Unity versions as well.

1. Import the LIV SDK into your project (Assets > Import > Custom Package).

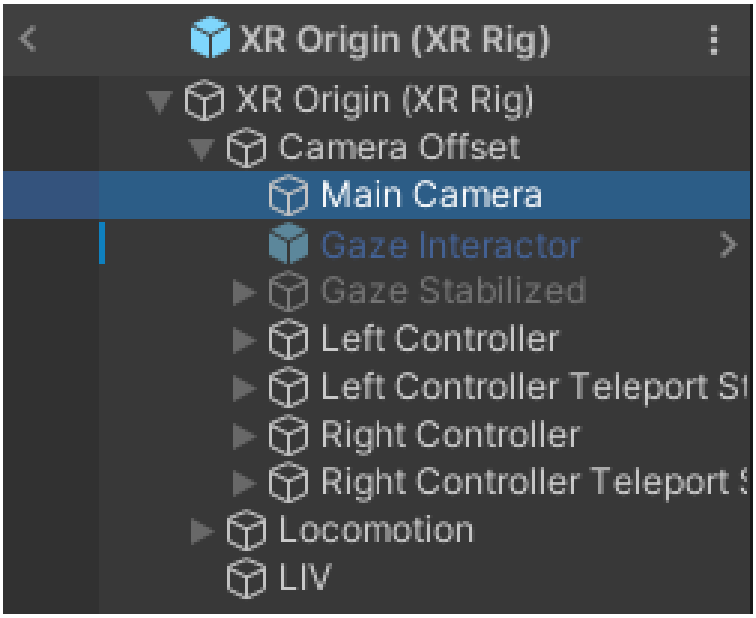


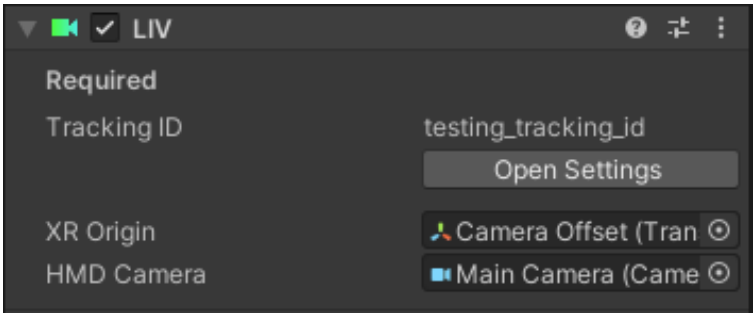
2. You can add the LIV SDK component anywhere in the project, but I add it as a parent of the **XR Origin (XR Rig)** prefab, so I don't have to add it to every scene manually. So I create an **Empty Child** to the root of *XR Origin (XR Rig)* prefab and add the **LIV script** to it.





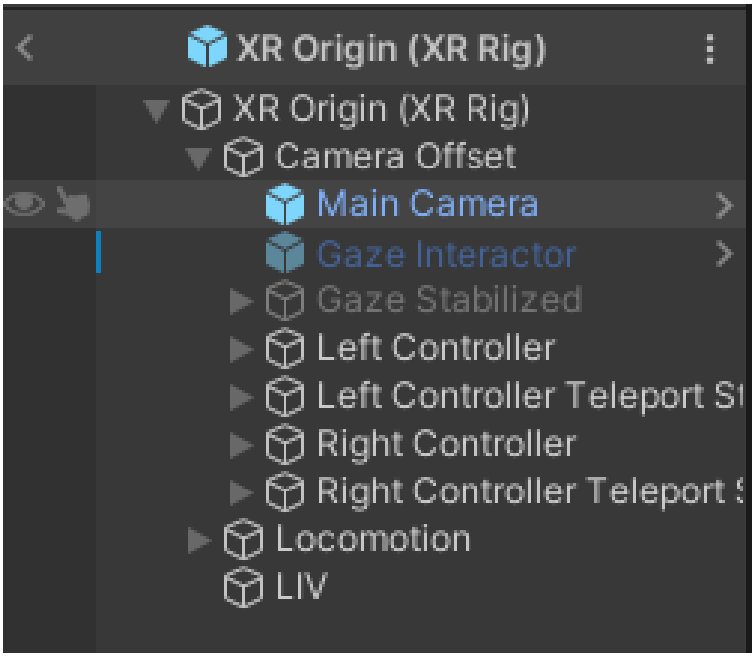
3. Next, we need to set the **HMD Camera** and the **XR Origin**. We will set the **HMD Camera** to the **Main Camera** in the **XR Rig**. We need to set the **XR Origin** component to the parent of **Main Camera** (i.e. **Camera Offset** in this case), and **NOT** the root parent of the prefab (i.e. **XR Rig**).

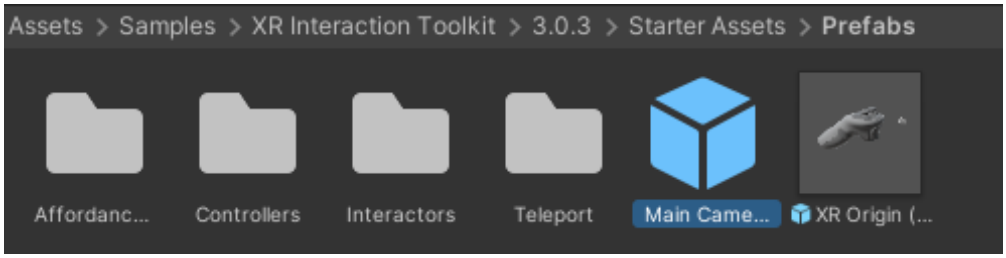




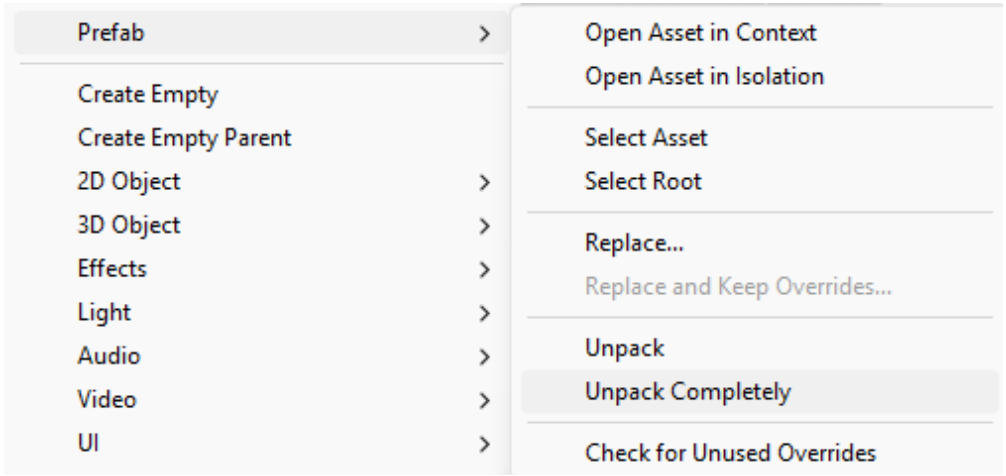
4. Now what we need to understand is that when we pass the Main Camera as the HMD Camera, LIV clones the camera and uses it for Mixed Reality Capture. But this might not be desirable if you have some components or some player specific code that should only run on the player controller (i.e. XR Rig), and not on the LIV Camera. So it is generally a better practice to create a separate prefab for the LIV camera, and remove any components that you might not want in the LIV camera. To do this:

    a. Drag the **Main Camera** to any folder in your Assets directory. This creates a prefab of the **Main Camera** GameObject:

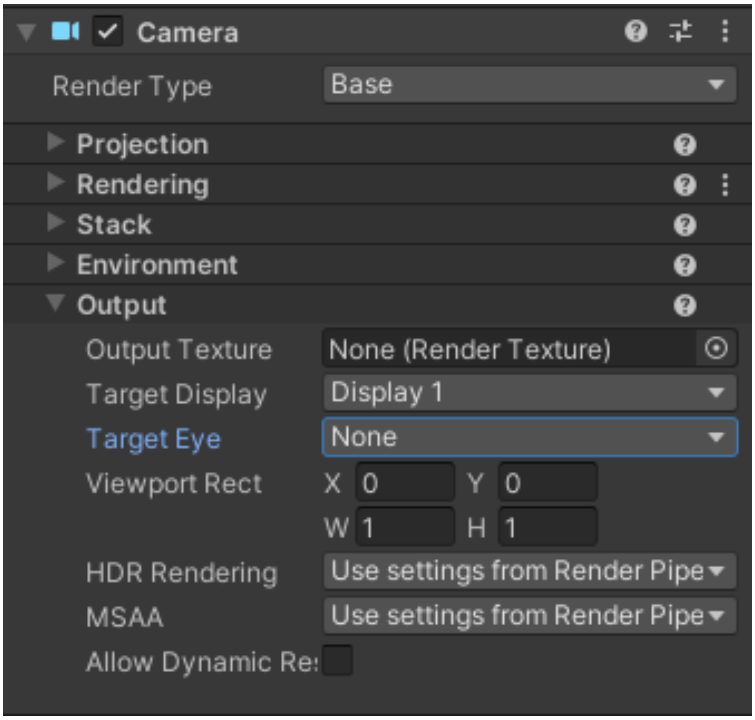




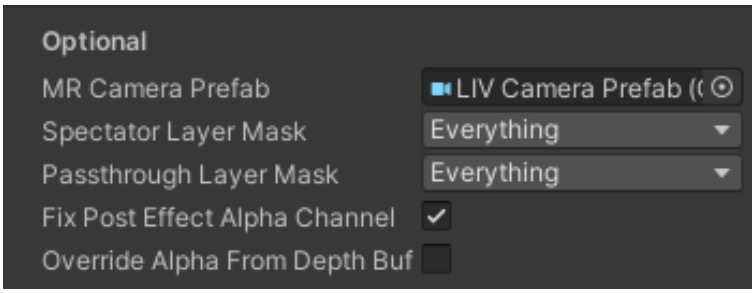
b. In the Hierarchy Window, select the **Main Camera** prefab and unpack it completely (Right-Click > Prefab > Unpack Completely).



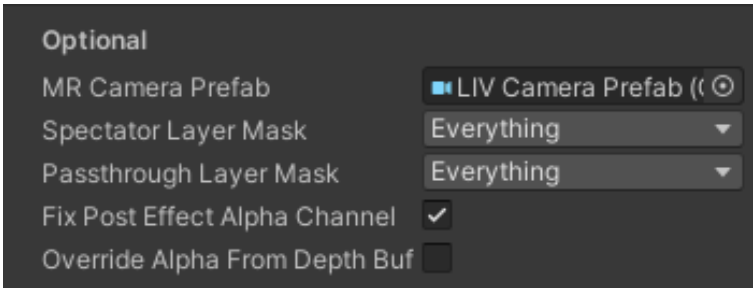
c. Rename the **Main Camera** Prefab to anything, *LIV Camera Prefab* in my case, and remove any unnecessary components or code from it. For example, set the **Target Eye** in **Camera Output** to **None** instead of **Both**, remove **AudioListeners**, etc.



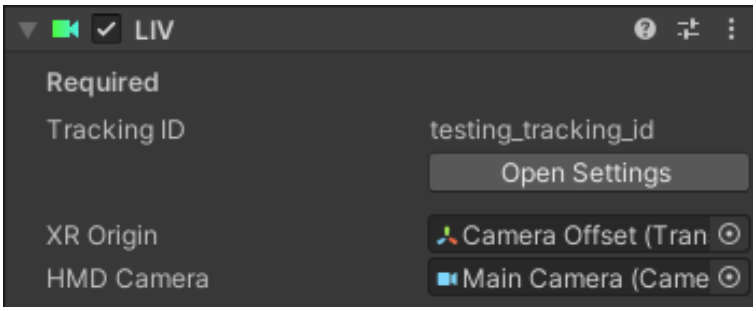
d. Now we can use it in the *MR Camera Prefab* field in the **LIV** Component:

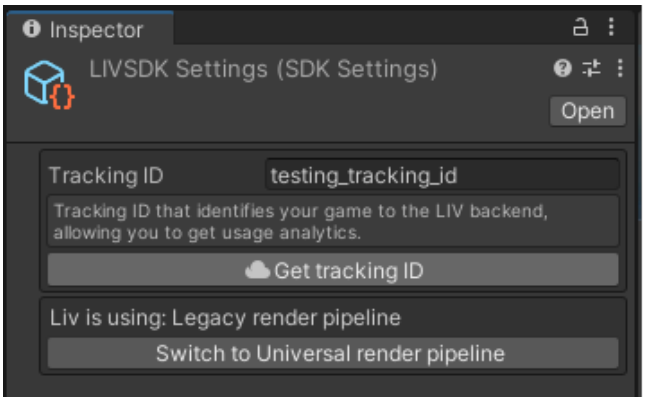


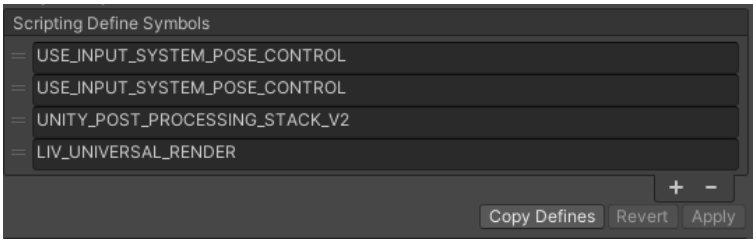
5. In the Spectator Layer Mask and Mask, you need to select all the layers that will be visible in your Virtual Camera and MR Passthrough respectively. You can just exclude the layers that you don't want to be visible in your LIV camera.



6. If you are using URP, you need to click *Open Settings* under LIV properties, and then **Switch to Universal Rendering Pipeline**. To confirm it is setup, go to (File > Build Settings... > Player Settings...) and scroll down until you see *Scripting Define Symbols* and check whether it contains **LIV_UNIVERSAL_RENDER**. If not, then add it manually using the + icon.



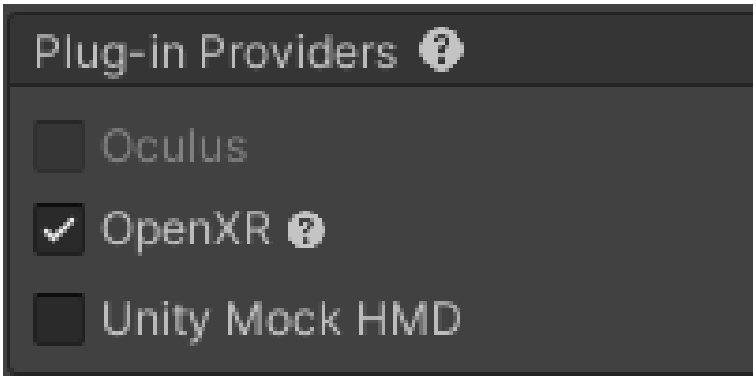




7. One thing to note is that **LIV** works with these runtimes:

   - SteamVR (OpenVR API)
   - OpenXR (When provided by SteamVR)

   So you need to use either the SteamVR API while building the project in Unity, or enable OpenXR under plugin providers in (Edit > Project Settings > XR Plug-in Management).



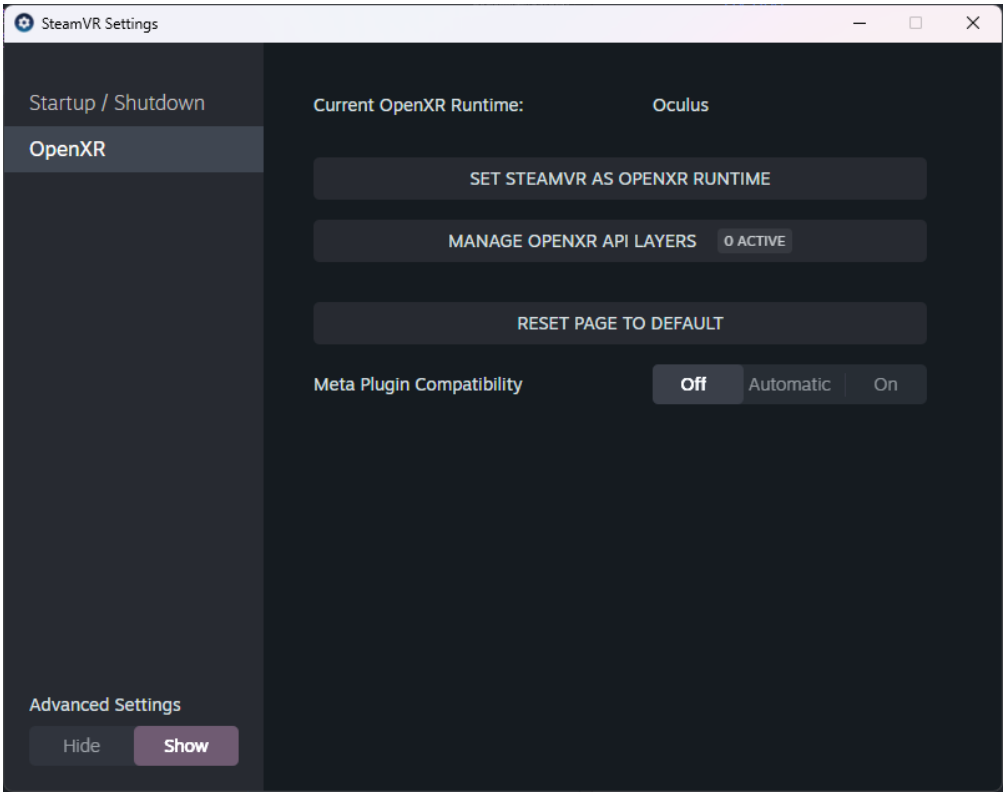
That is all for integrating LIV SDK into Unity. For additional information, look at the official documentation for LIV SDK. Onto testing the integration now.

1.4 LIV Installation and Setup

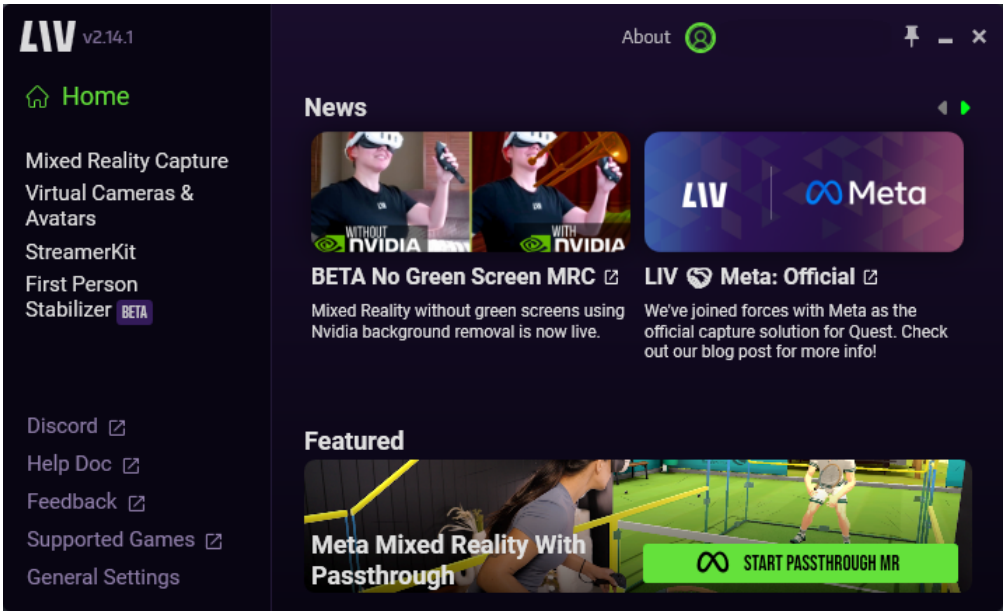
First thing you need to do is install these applications from Steam.

- SteamVR
- LIV App

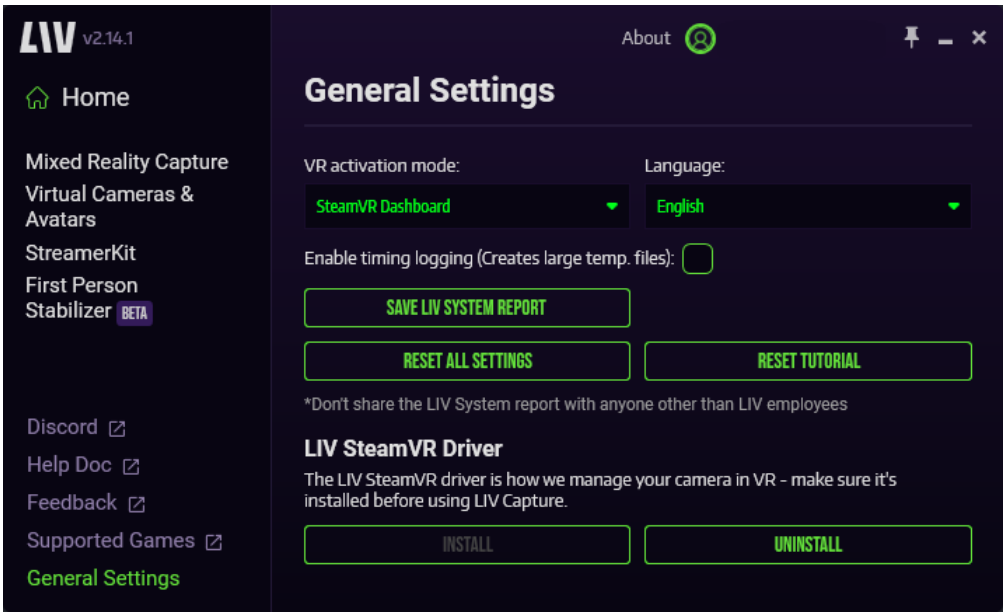
Now, start *SteamVR* and set the OpenXR runtime to SteamVR by going to (Settings > OpenXR > Set SteamVR as OpenXR Runtime).



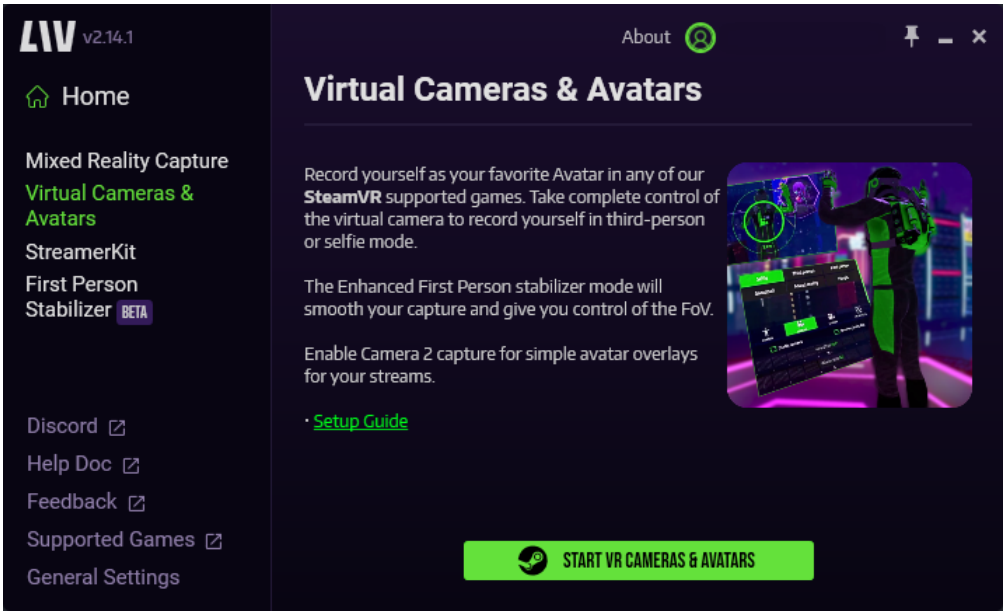
Now put on your headset and connect through Meta Link (in case of Oculus headsets).
After that, launch LIV through Steam.



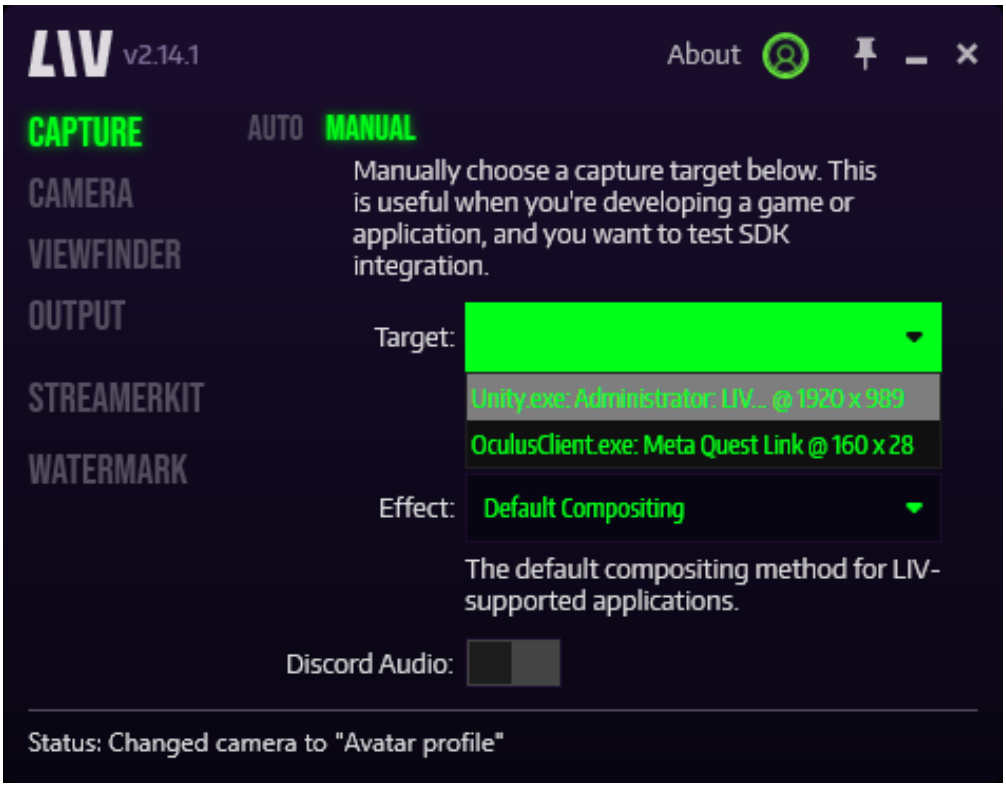
Go to *General Setting*, and click on **INSTALL** under *LIV SteamVR Driver*.



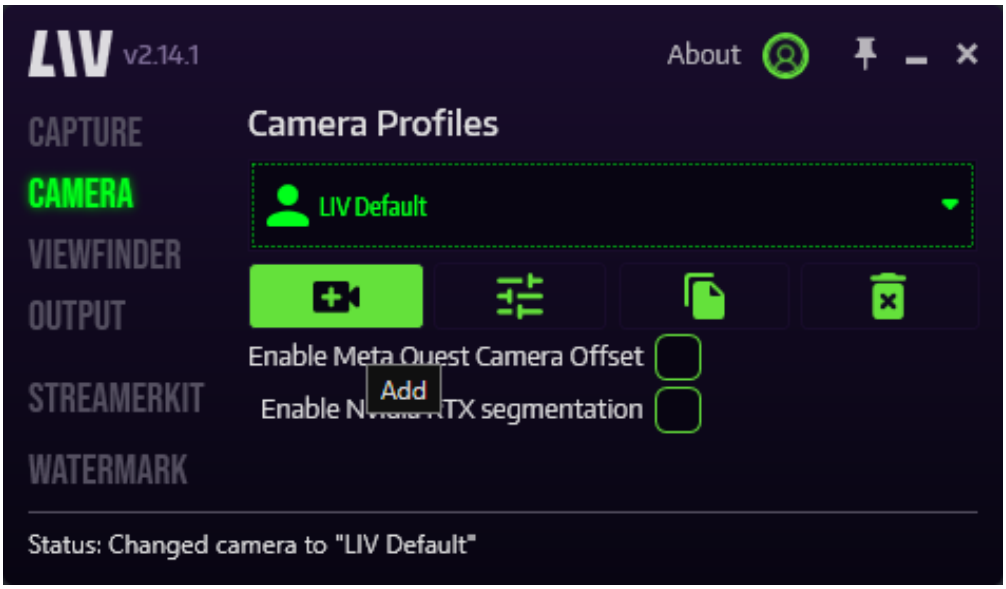
After that, go to *Virtual Cameras & Avatars* and click **START VR CAMERAS & AVATARS**.



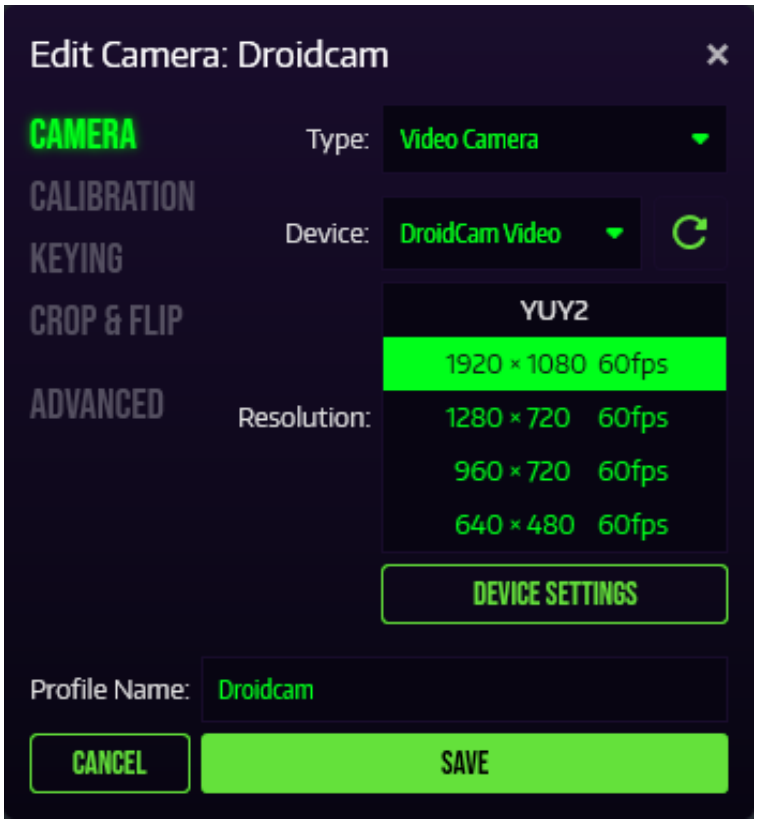
Go to (Capture > Manual) and select the Executable of your game if you are running a build, or select *Unity.exe* if you're running it through Unity in **Play Mode**.



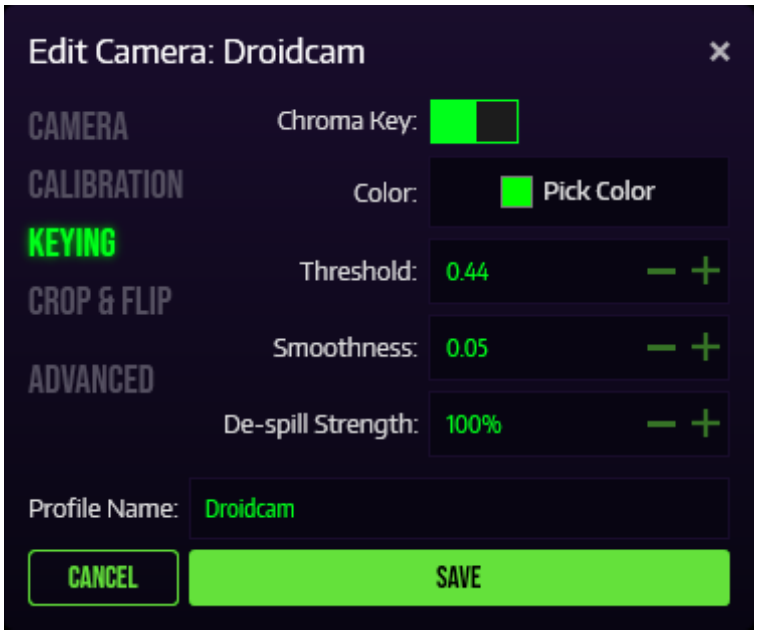
Go to *Camera* and click on **Add** to create a new camera profile.



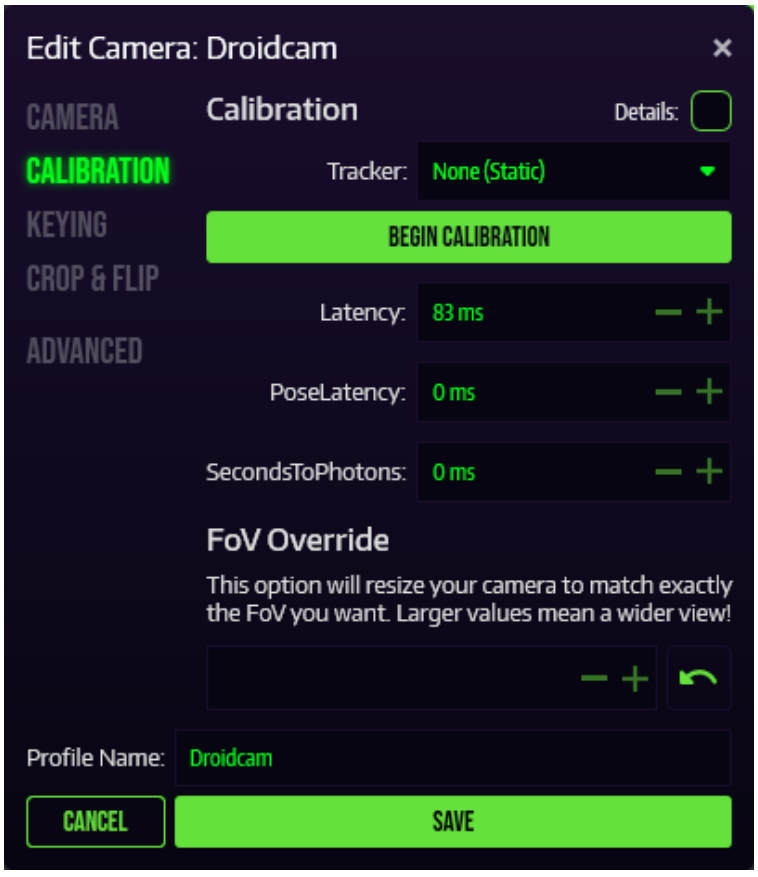
Set the Type to *Video Camera*, select your camera device under *Device* (DroidCam Video in my case), and select the resolution you want.



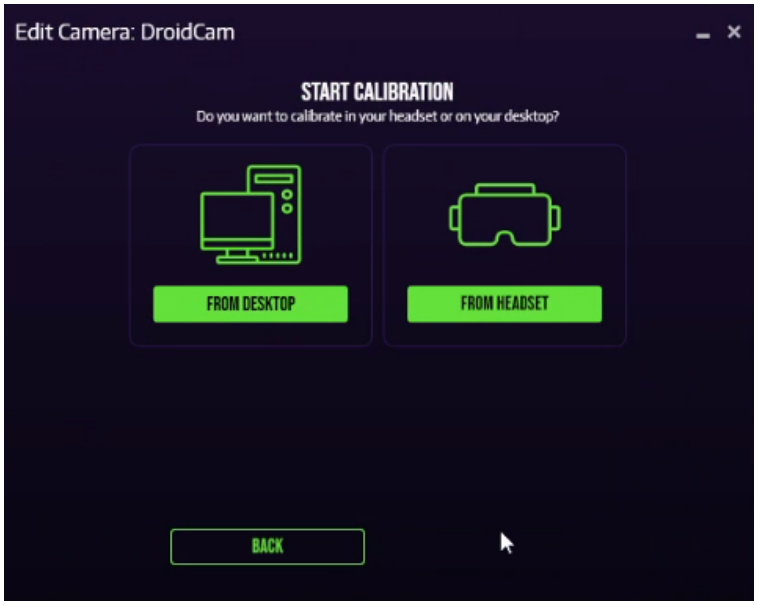
Go to *Keying* and select the color you want to **Chroma Key**, its threshold, smoothness, and de-spill strength.



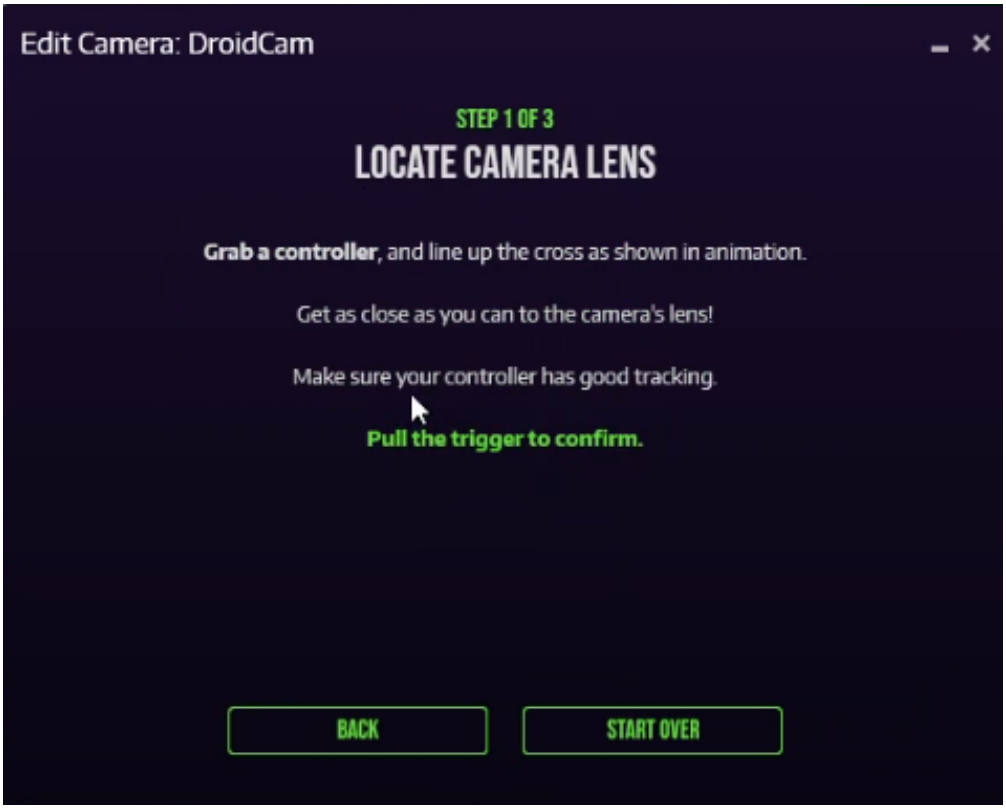
Next, go to *Calibration* and click on **Begin Calibration**.



Select *Desktop* and now put on your VR headset.

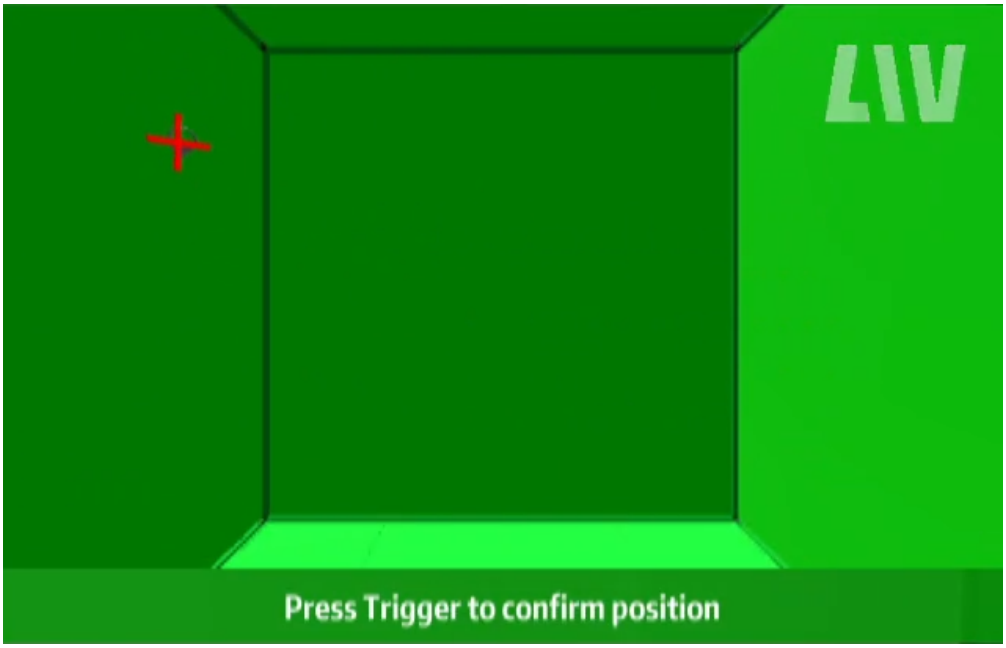


Now you are in calibration mode for the camera. Remember that since we do not have a tracker for the camera, we cannot move the camera, and every time we move the camera we have to re-calibrate the camera.

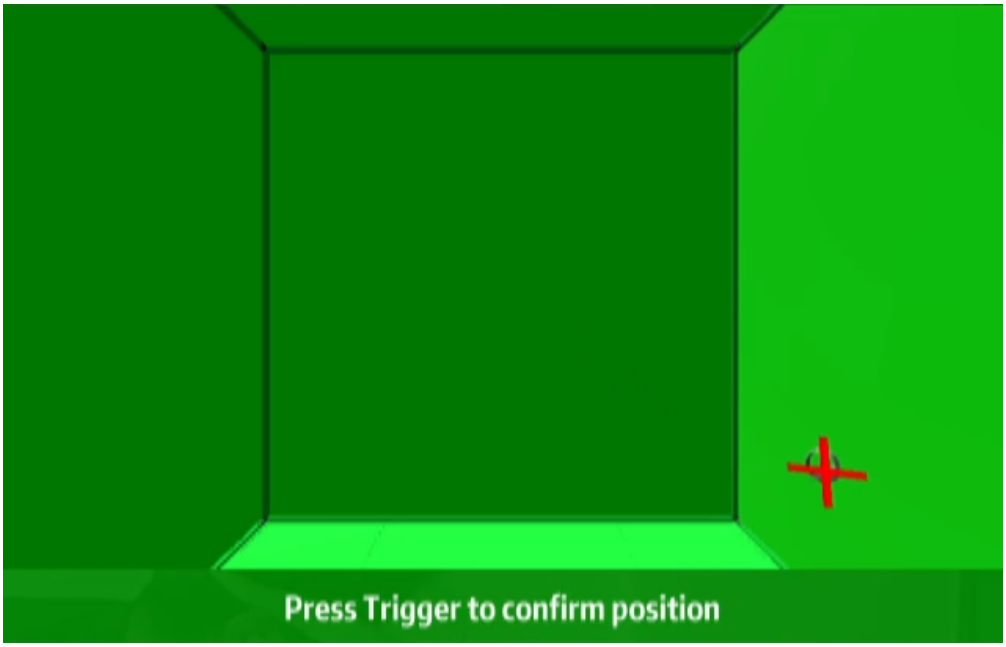


In the **LIV Output** window, you will be able to see the markers. You have to follow the instructions on the screen corner and place your controller as close to the camera as possible, then press the trigger button.

Then for the second marker, move back a bit, go to the corner on the screen shown in the instructions, align your controller with the marker, and press the trigger button.



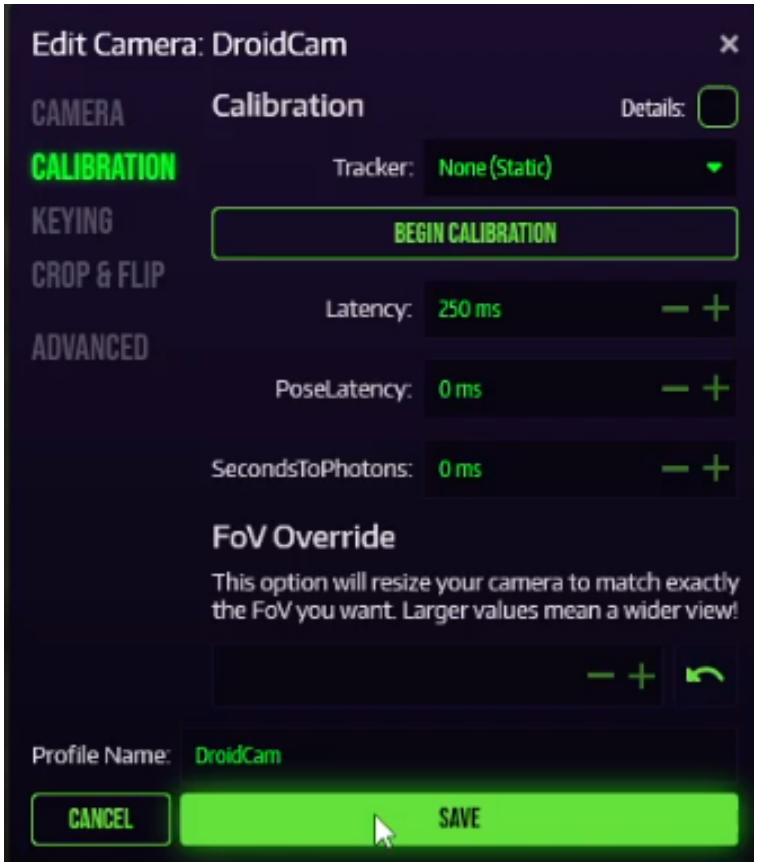
Similarly for the third marker, align the controller with the marker and press the trigger button.



After that, you will see virtual controllers that are aligned to the controllers from the camera video, and follow the real controllers.

You can also adjust the latency of the virtual controller movement to ensure they are in sync with the controllers in the video feed. Adjustments to the X, Y, and Z position and rotation of the controller can be made manually from within the LIV app in the headset for micro adjustments.

Now you can save the camera settings and LIV is ready to record the content from the game.


Edit Camera: DroidCam
CAMERA
CALIBRATION
KEYING
CROP & FLIP
ADVANCED
Calibration
Details:
Tracker:
None (Static)
BEGIN CALIBRATION
Latency:
250 ms
PoseLatency:
0 ms
SecondsToPhotons:
0 ms
FoV Override
This option will resize your camera to match exactly the FoV you want. Larger values mean a wider view!
Profile Name:
DroidCam
CANCEL
SAVE