

Quick start guide

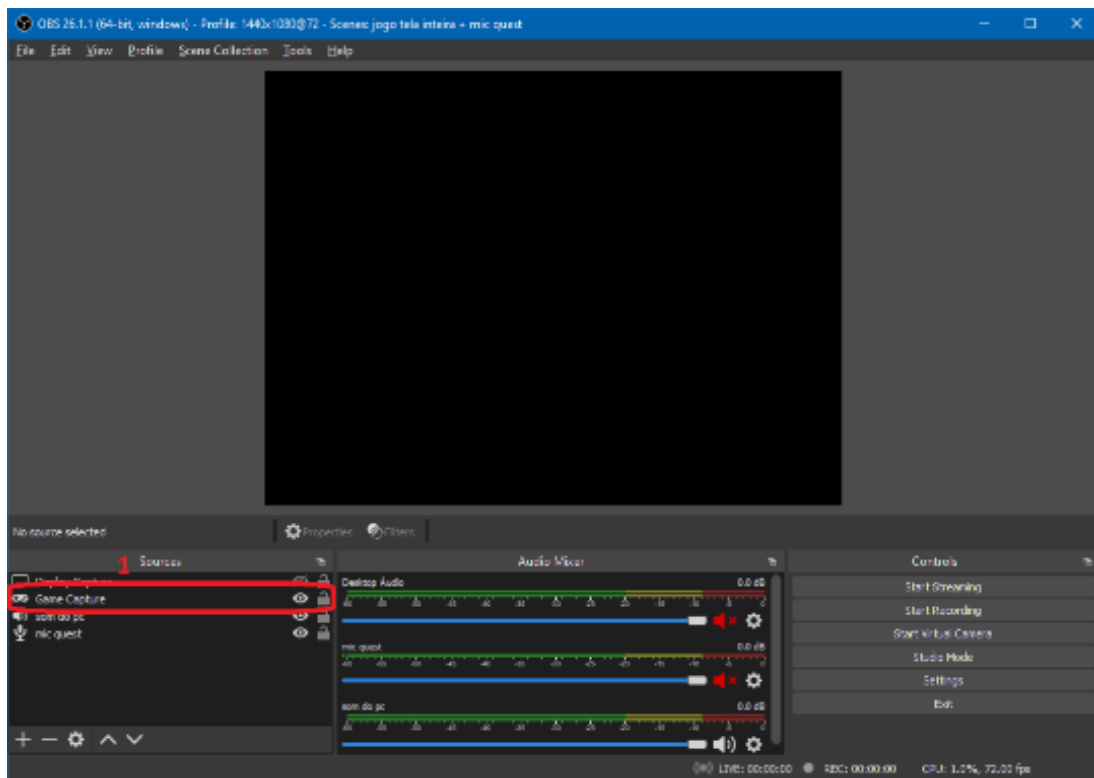
This script requires avisynth (free); it can be installed automatically by request on the first execution, manually through the avisynth_installer folder or manually downloaded at:

https://github.com/AviSynth/AviSynthPlus/releases/download/v3.7.0/AviSynthPlus_3.7.0_20210111.exe

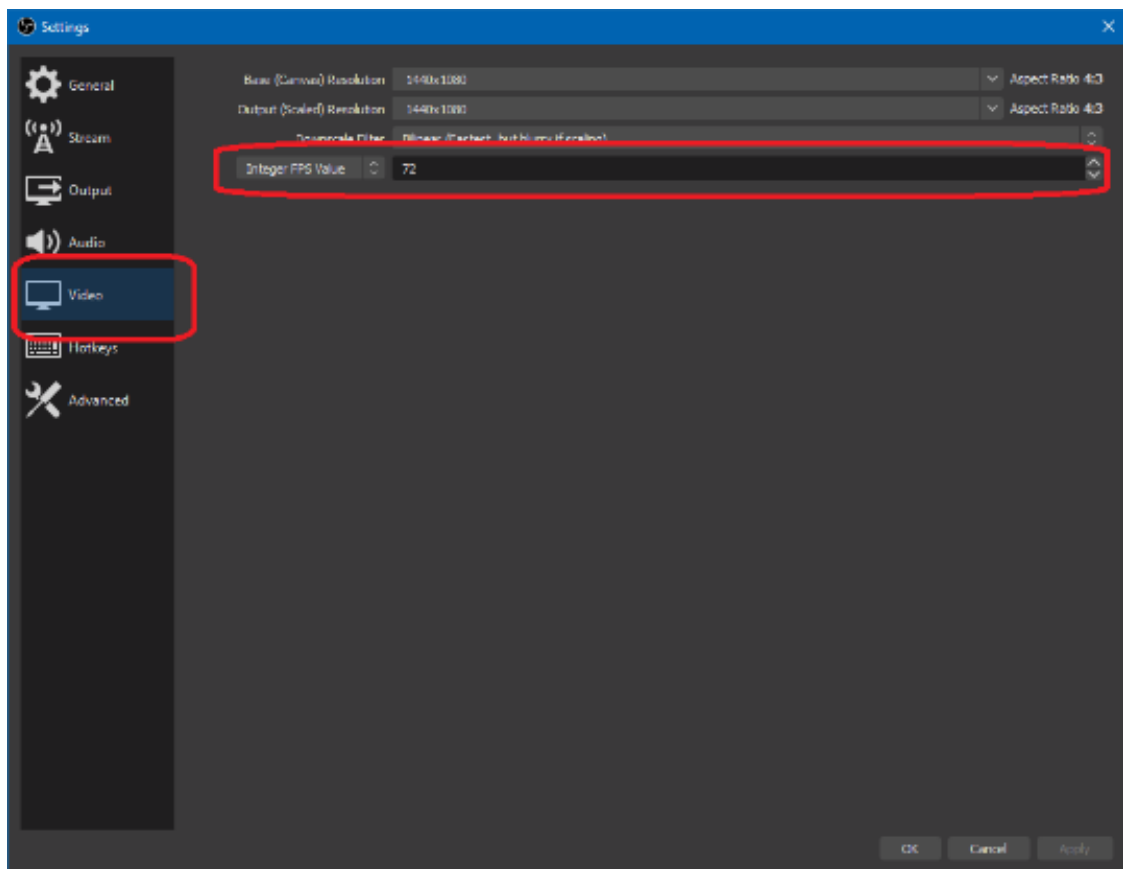
You are going to need a good source video file to use, **ideally recorded at the same frame rate the game is running on your VR device**, so avoid using nvidia shadowplay (it records at just 60fps). The regular refresh rates for VR devices are in the range of 72hz (low end) to 144hz (high end).

A second consideration is that a **wider field of view capture (FOV) is often better**, because for the video to be deshaken it needs some spare room to be “zoomed in”. The simplest solution is to run your VR game in a window that you can resize and reshape to create a 4:3 display. Not all games play along well and some require .ini tweaks or alt+enter to run windowed.

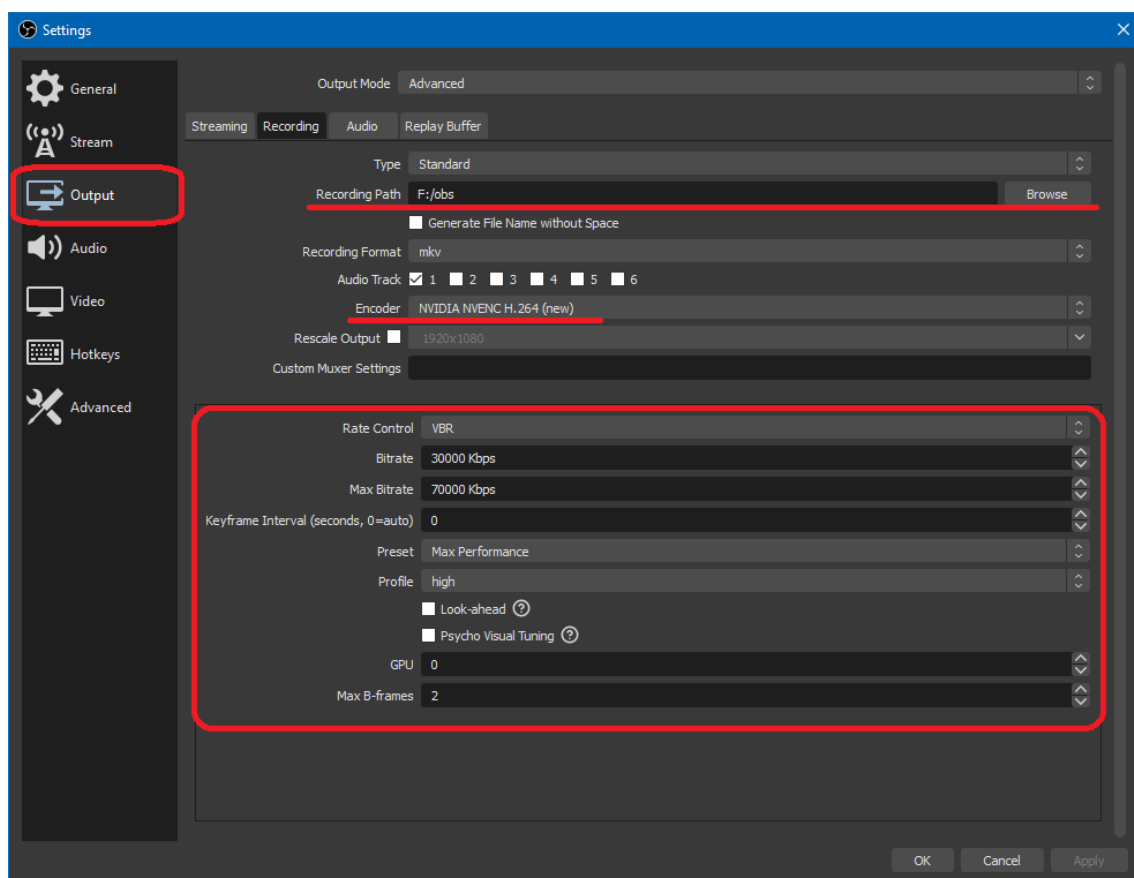
A strong suggestion is to use OBS (free, <https://obsproject.com>) to record your footage, with the following:



Use a **gamecapture source** if you are going to record directly from the game window. Desktop capture might be limited by the refresh rate of your monitor (60hz?) and will cause “bad framepacing” in the video.



Capturing the game at the **same frame rate your VR device screen Hz is running** is very important for fluid motion in the result!



If you are not already familiar with OBS, in this tab you will select your local record settings like bitrate, what encoder to use (nvenc h264 if you have a nvidia card) and some other tweaks that are covered in detail by other guides on the internet.

Game window shape and its relation with field of view (FOV)



On the left we have a narrow regular sized 16:9 FOV, and in the right a wide vertical FOV. If you record your video in the 16:9 format (left) you might lose visible detail that is being cut out of the frame on the top and bottom. You might want to fiddle with the window size (if possible) to figure out what works best for you. My recommendation is “wider vertical, 4:3 ratio”.

What about video captured directly from the Oculus Quest?

It is a very mixed bag. By default, the settings are 1024x1024 resolution (good fov) but 30 fps (very bad framerate).

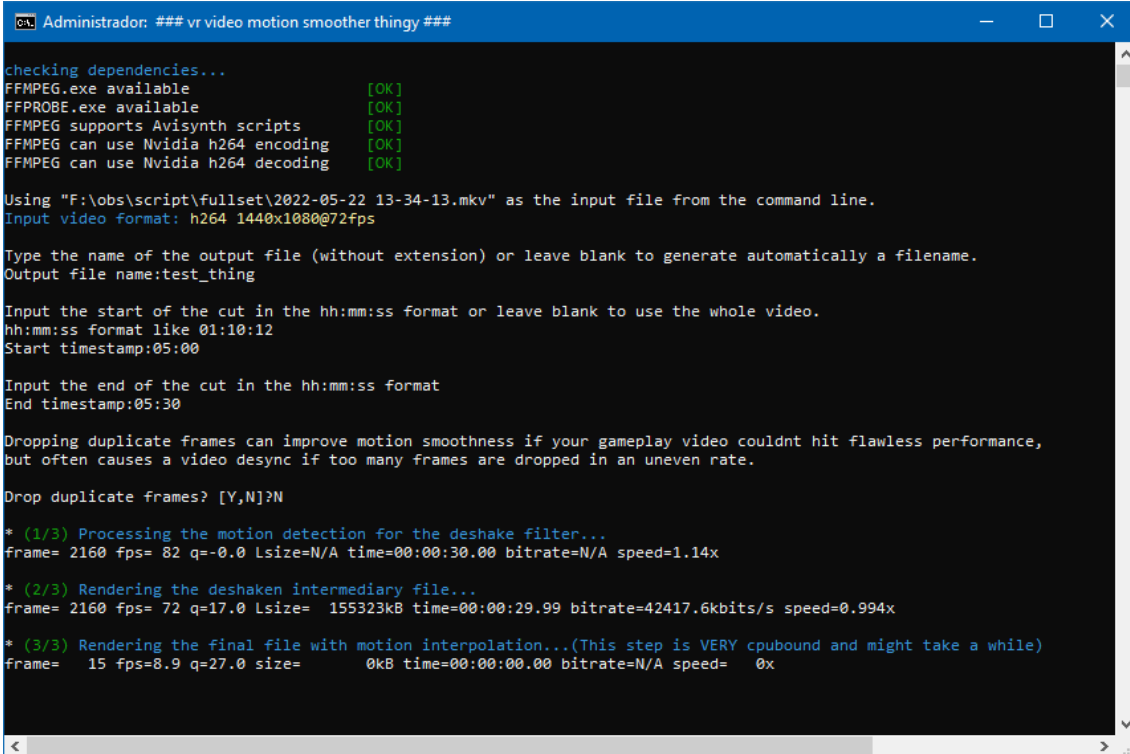
If you have the developer mode enabled, you can try to tweak the recording quality up matching the screen refresh rate or exactly half or it. However, those aren't officially supported and may affect your game performance, with causes other problems on their own.

Guide describing how to tweak those settings: <https://www.roadtovr.com/increase-video-capture-resolution-quest-2-recording-quality/>

Ps: I will call it Oculus Quest till the end of the days, bite me meta. =P

Running the video conversion tool

The script when executed looks like this, it is straightforward to use.



```
Administrador: ### vr video motion smoother thingy ###

checking dependencies...
FFMPEG.exe available [OK]
FFPROBE.exe available [OK]
FFMPEG supports Avisynth scripts [OK]
FFMPEG can use Nvidia h264 encoding [OK]
FFMPEG can use Nvidia h264 decoding [OK]

Using "F:\obs\script\fullset\2022-05-22 13-34-13.mkv" as the input file from the command line.
Input video format: h264 1440x1080@72fps

Type the name of the output file (without extension) or leave blank to generate automatically a filename.
Output file name:test_thing

Input the start of the cut in the hh:mm:ss format or leave blank to use the whole video.
hh:mm:ss format like 01:10:12
Start timestamp:05:00

Input the end of the cut in the hh:mm:ss format
End timestamp:05:30

Dropping duplicate frames can improve motion smoothness if your gameplay video couldnt hit flawless performance,
but often causes a video desync if too many frames are dropped in an uneven rate.

Drop duplicate frames? [Y,N]>N

* (1/3) Processing the motion detection for the deshake filter...
frame= 2160 fps= 82 q=-0.0 Lsize=N/A time=00:00:30.00 bitrate=N/A speed=1.14x

* (2/3) Rendering the deshaken intermediary file...
frame= 2160 fps= 72 q=17.0 Lsize= 155323kB time=00:00:29.99 bitrate=42417.6kb/s speed=0.994x

* (3/3) Rendering the final file with motion interpolation...(This step is VERY cpubound and might take a while)
frame= 15 fps=8.9 q=27.0 size= 0kB time=00:00:00.00 bitrate=N/A speed= 0x
```

It will ask you for the input file, I recommend dragging the file to the window to autofill this field and then press enter.

Then it will ask for a output file name to save the result, you can leave it blank to auto generate a filename on the same folder as the script.

It will ask for the time frame cut of the main video, something like from 01:03:40 to 01:05:00 in the hh:mm:ss format. You can leave this blank to use the whole file.

It will ask you if you want to drop duplicate frames during the conversion, if you pick yes it might result in a much cleaner motion if your game didn't ran with perfect performance during the capture, but also it may desync the audio if too many frames where dropped.

The output file is always 60fps, created by a motion-interpolated down sampling mixing all the available frames from the original framerate. Bigger input frame rates (like 144fps) will generate less motion artifacts in the processed video, but 72hz is already very good. (I am a quest 1 user)

Author: Aruã Metello - Contact: aruametello@gmail.com