Mattatz's volume renderer (based on Kyle Hayward's mathematics), while relatively undeveloped compared to its paid Unity asset store counterparts, is lightweight enough to shipped to Android and iOS devices, and runs quickly with low load times (as long as the imported dcm is not excessively large).

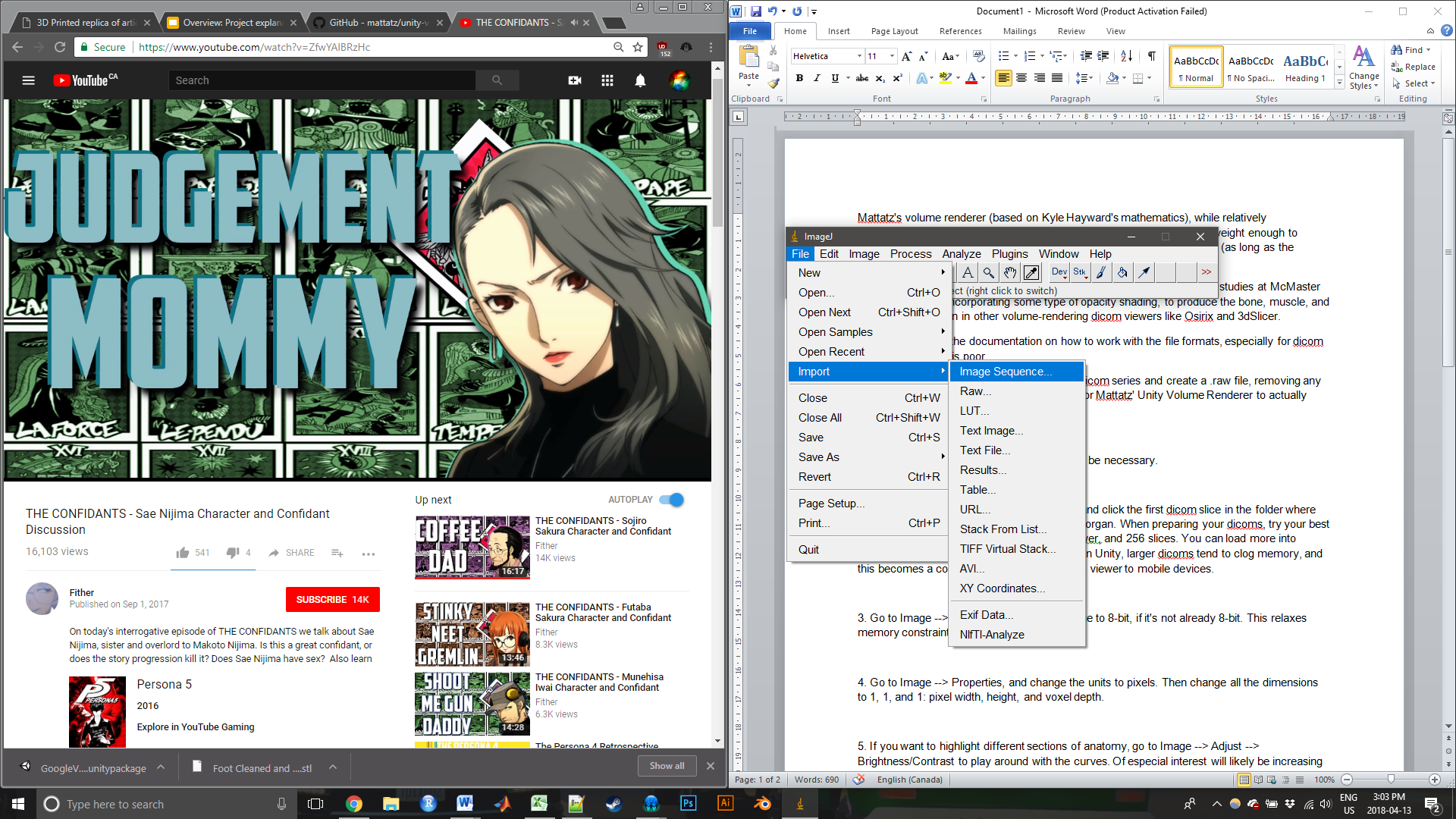
Future implementations of Mattatz' renderer for the purposes of Anatomy studies at McMaster may even involve incorporating some type of opacity shading, to produce the bone, muscle, and vessel coloring seen in other volume-rendering dicom viewers like Osirix and 3dSlicer.

However, some of the documentation on how to work with the file formats, especially for dicom and Unity novices, is poor.

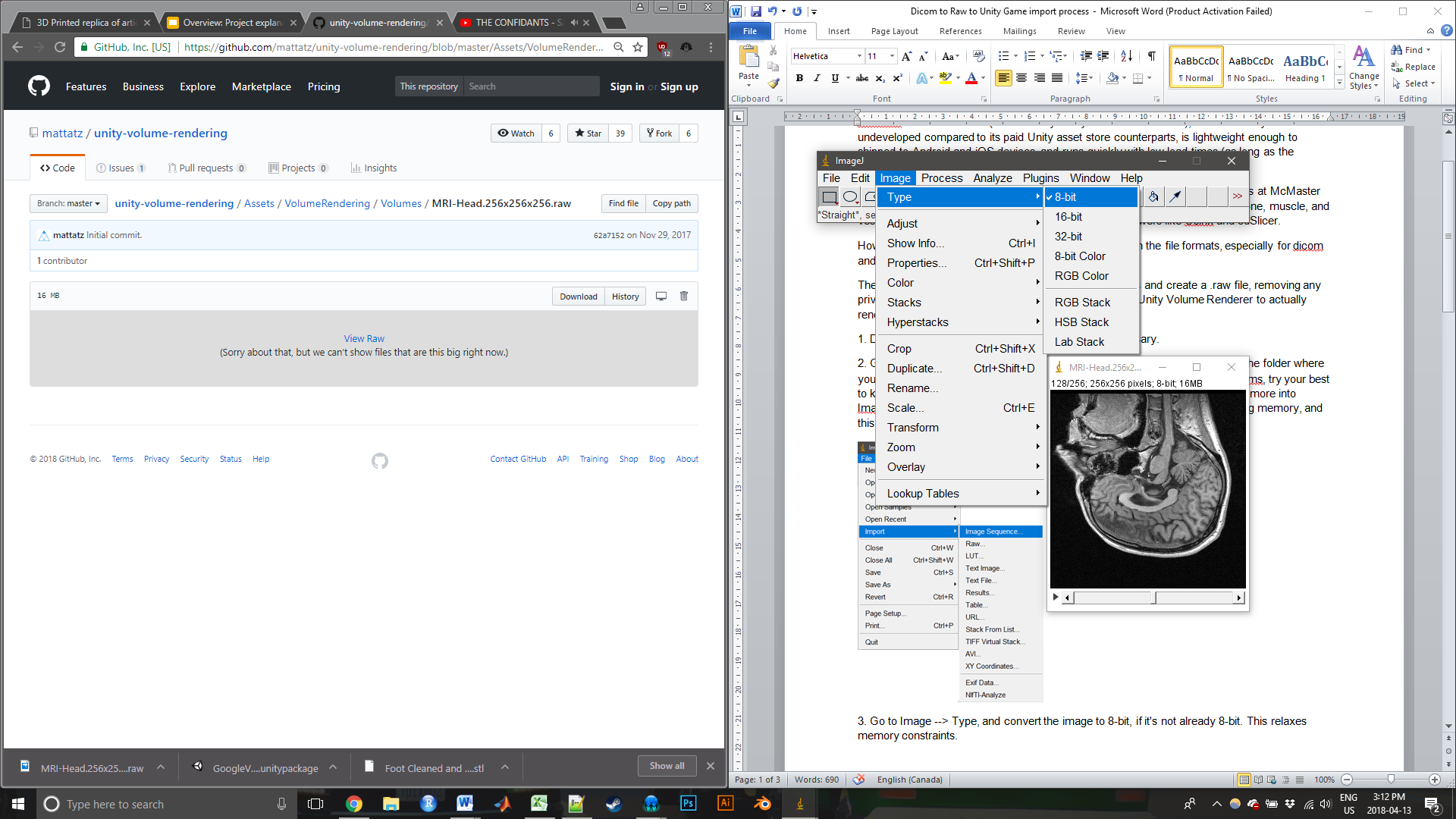
The following is a series of steps to take any dicom series and create a .raw file, removing any private information, and creating an input file for Mattatz' Unity Volume Renderer to actually render.

1. Download **ImageJ**. No extra plug-ins should be necessary.

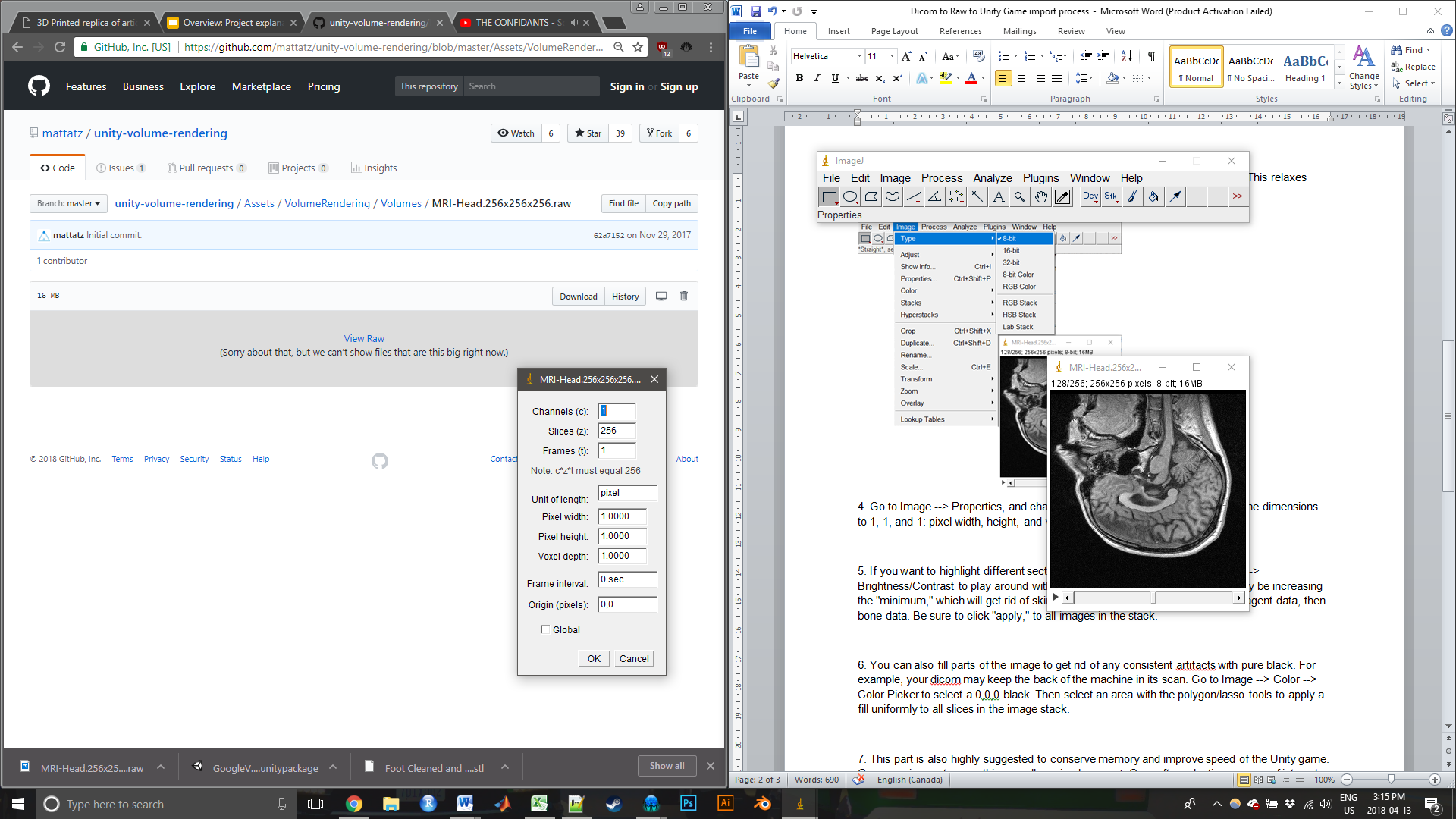
2. Go to **File** 🡪 **Import** 🡪 **Image Sequence**, and click the first dicom slice in the folder where you stored your batch of slices for a particular organ. When preparing your dicoms, try your best to keep the image resolution to 256x256 or lower, and 256 slices. You can load more into ImageJ, but when you export the .raw for use in Unity, larger dicoms tend to clog memory, and this becomes a concern when shipping the VR viewer to mobile devices.



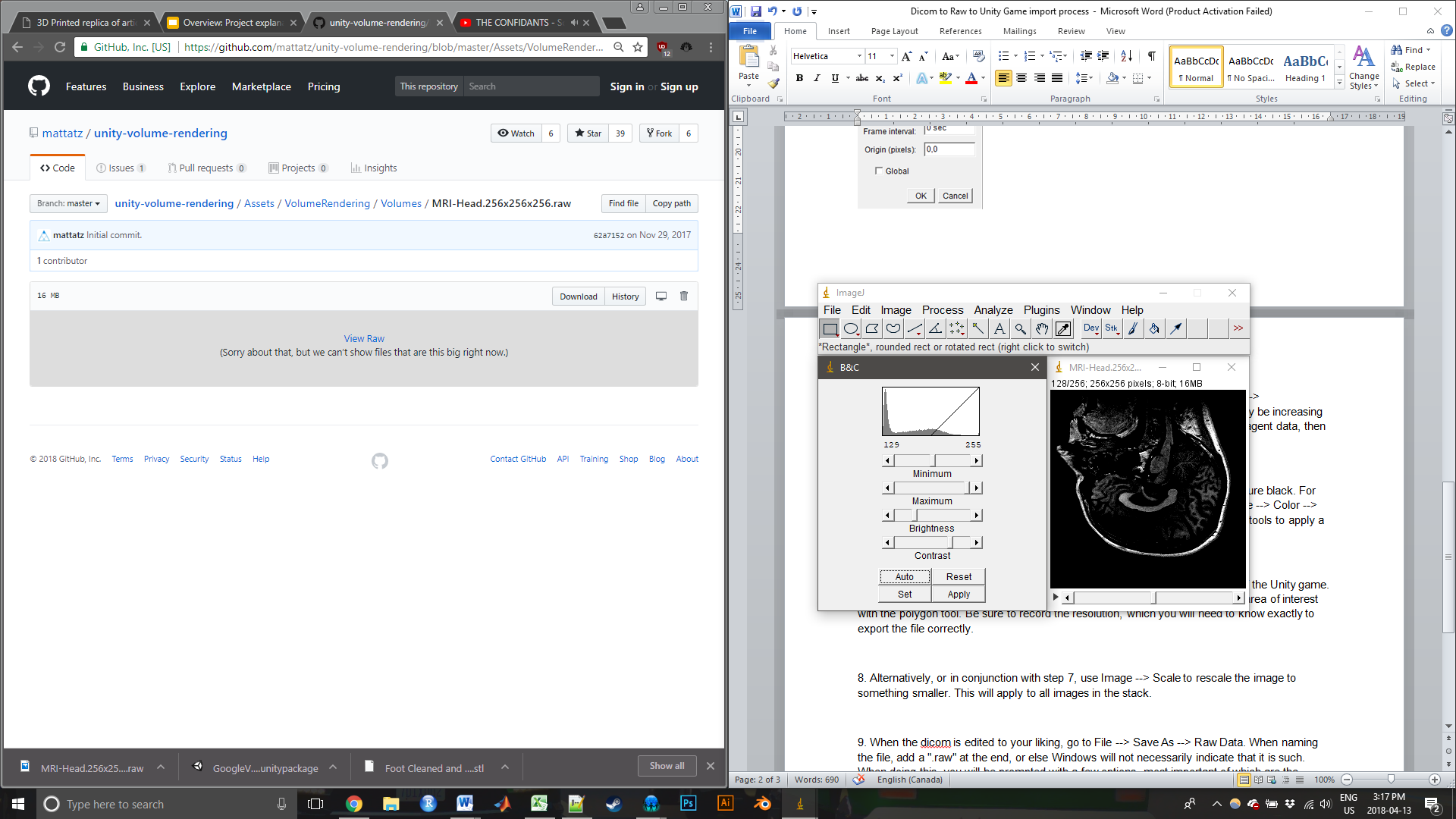
3. Go to **Image 🡪 Type**, and convert the image to 8-bit, if it's not already 8-bit. This relaxes memory constraints.



4. Go to **Image** 🡪 **Properties**, and change the units to pixels. Then change all the dimensions to 1, 1, and 1: pixel width, height, and voxel depth.



5. If you want to highlight different sections of anatomy, go to Image 🡪 Adjust 🡪 Brightness/Contrast to play around with the curves. Of especial interest will likely be increasing the "minimum," which will get rid of skin/fat/muscle data, then BriteVU contrast agent data, then bone data. Be sure to click "apply," to all images in the stack.



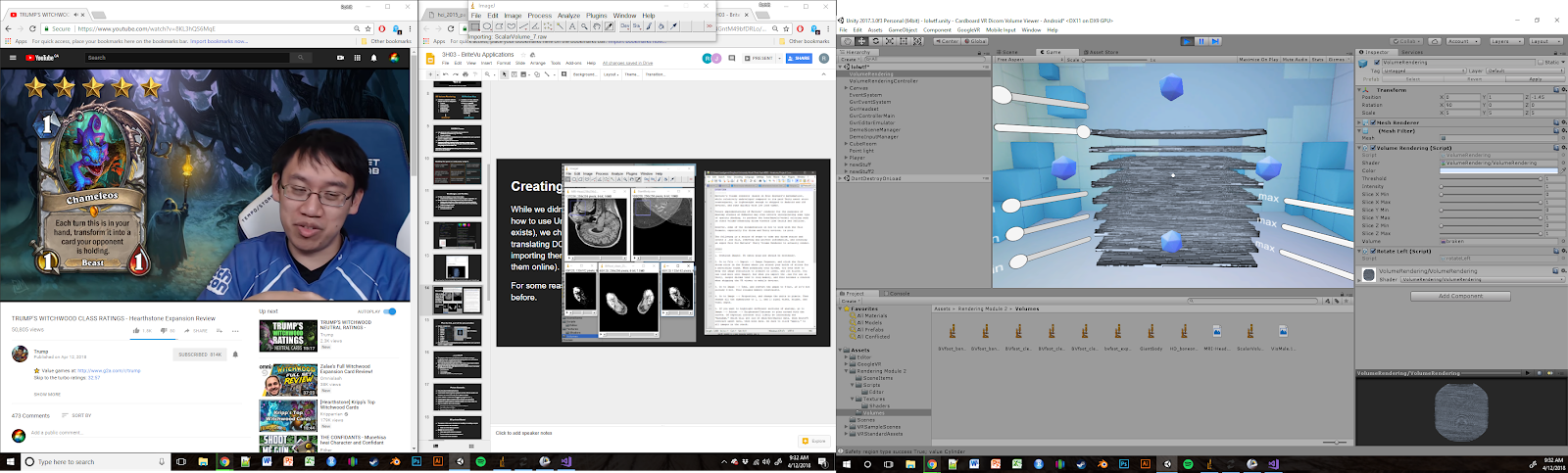
6. You can also fill parts of the image to get rid of any consistent artifacts with pure black. For example, your dicom may keep the back of the machine in its scan. Go to **Image 🡪 Color 🡪 Color Picker** to select a 0,0,0 black. Then select an area with the polygon/lasso tools to apply a fill uniformly to all slices in the image stack.

7. This part is also highly suggested to conserve memory and improve speed of the Unity game. Crop your image to something smaller using **Image 🡪 Crop** after selecting an area of interest with the polygon tool. Be sure to record the resolution, which you will need to know exactly to export the file correctly.

8. Alternatively, or in conjunction with step 7, use **Image 🡪 Scale** to rescale the image to something smaller. This will apply to all images in the stack.

9. When the dicom is edited to your liking, go to **File 🡪 Save As 🡪 Raw Data**. When naming the file, add a ".raw" at the end, or else Windows will not necessarily indicate that it is such. When doing this, you will be prompted with a few options, most important of which are the resolution of each image, and the number of slices. The number of slices, if too high or too low, will not ruin the file -- you will merely export only the first n images. If the resolution is wrong, the file is corrupted. Be sure to input the correct resolution.

10. Once imported into Unity, go to **Window 🡪 Editor**, and use the custom editor script, "VolumeAssetBuilder." Input the .raw file you imported into the window that pops up. **Input the correct resolution and slice count. If the input is wrong, or at any point the right unit is not used to describe the pixel size and voxel depth in step 4, the renderer will fail.**



11. Once the .asset file is made, select the VolumeRendering gameObject in the scene, and in the Volume input under Volume Rendering (Script) in the Inspector tab, input the .asset file.

12. Test the game, and if it is to your liking, build the game and test it on mobile.