legends: Orange (Medium Priority), Red (Critical/High Priority), Blue (Info), Green (Good work!)

Done, Important

# Monday, 15 July 2024

4th Meeting with Dr Hansung Kim @ 10.30am

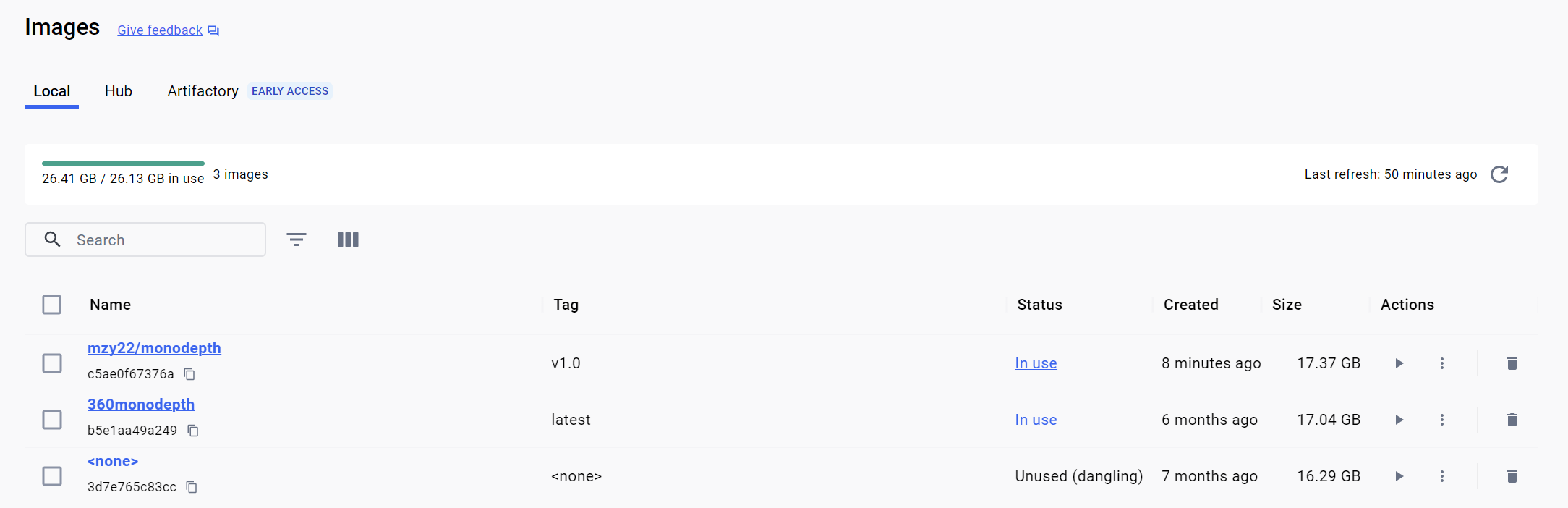
* Keep working in perfecting Unity project to make it as realistic and immersive
* Non VR mode player controller should have collision
* Binaural audio for demonstration
* Mono audio is fine for RIR. (hmm now I think about it, maybe not fine but needed, need to follow how real RIR are calculated, check mic/speaker config for ground truth)
* Mona leaving for WFH starting next week, so ask whatever you need now!
* Email Dr Hansung with recording for feedback.

**UPDATED TODO:**

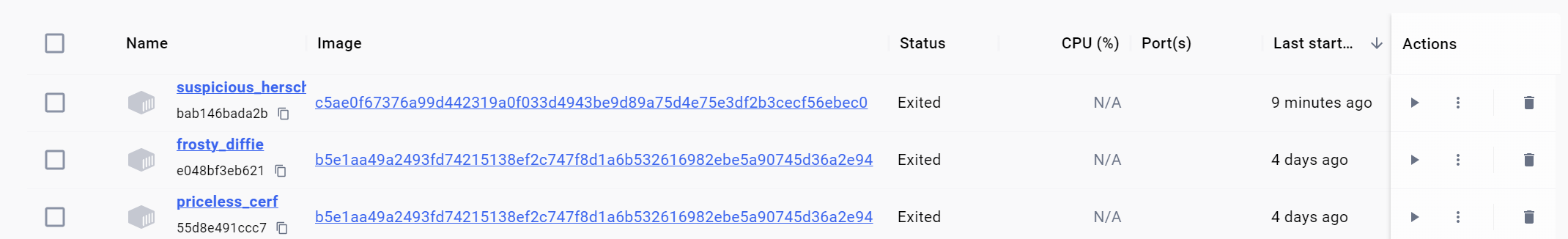
1. Figure out Steam Audio quirks and revamp Unity project
   1. Ask Mona for old unity project for reference, and also for positions of probe/listener/source.
   2. Get MATLAB code for RT60 and EDT calculation.
2. Continue monodepth optimisation ( fix bug on using boostingMonocularDepth on 360monodepth), also check if GDP’s original folder have the submodule or not to make sure.
3. Remove docker clutter
4. Remove LiDAR from LFS, and use separate cloud service to download that, or use MeshLab to optimize it.

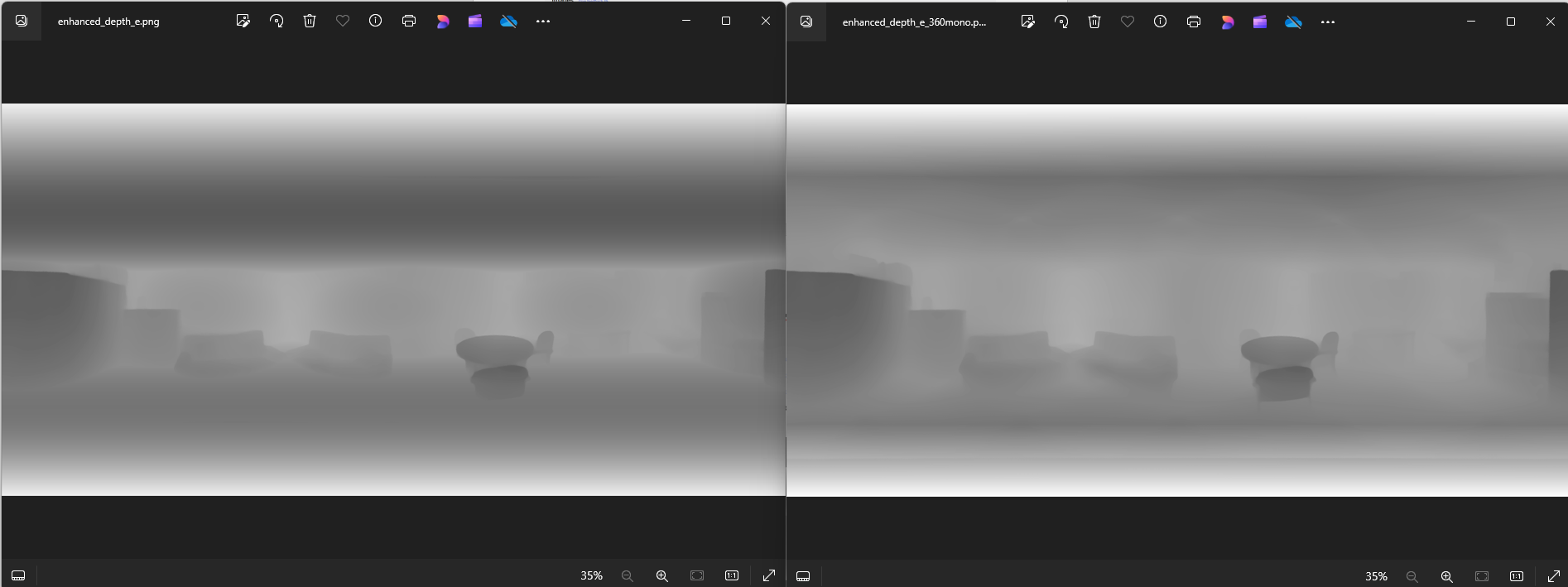
After rereading GDP report on Evaluation part specifically, it also mentions about the limitation from using 360monodepth with edgenet360 which is built and trained with stereo depth map. However, after looking into Future Work section, it seems, they really didn’t implement BoostingMonocularDepth for some reason. This means I should try it. Note to myself: check back the gdp report now and then on specific part to double check their intention and their finding, especially in methodology, evaluation and future work. Don’t be stupid and research same thing and get same finding, but it is good idea to reconfirm their findings!

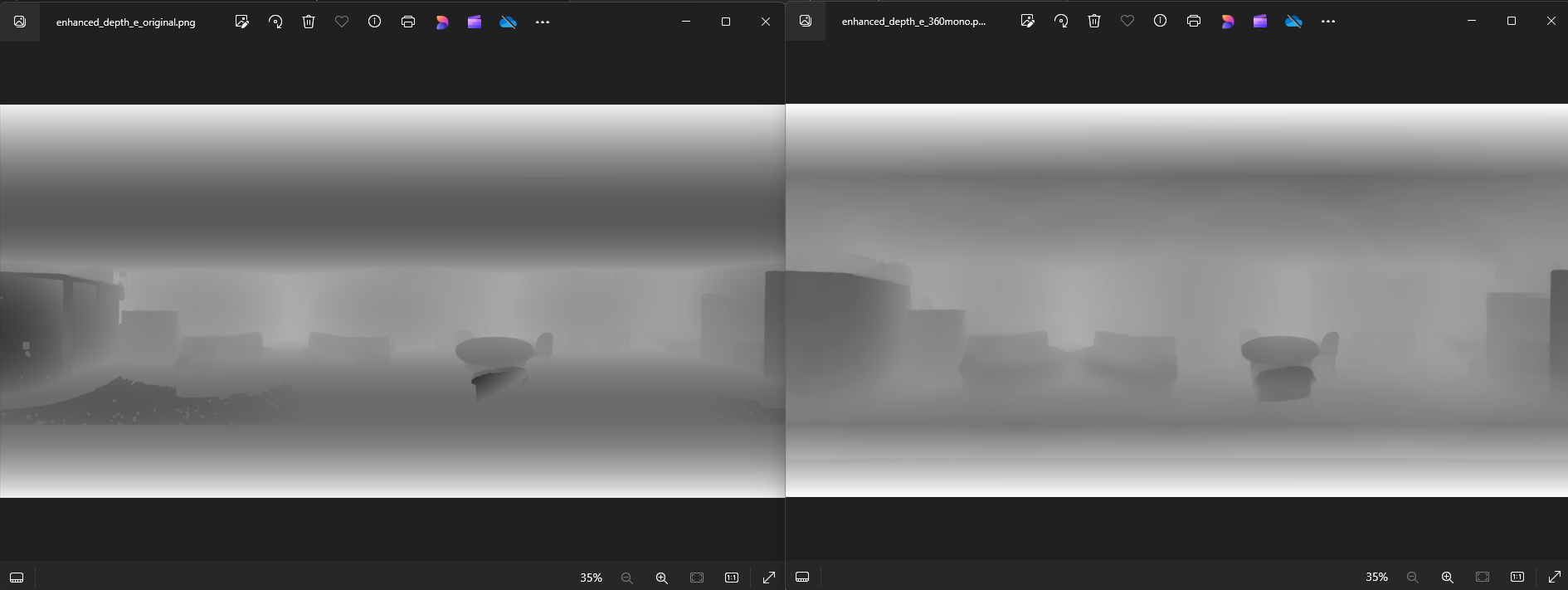
Fixed dockerfile build error on line 39 by replacing it with ‘RUN if [ -d "BoostingMonocularDepth" ]; then echo "Submodule exists"; else echo "Submodule not found" && exit 1; fi’ to check if submodule already exist (manually update before running docker build.)

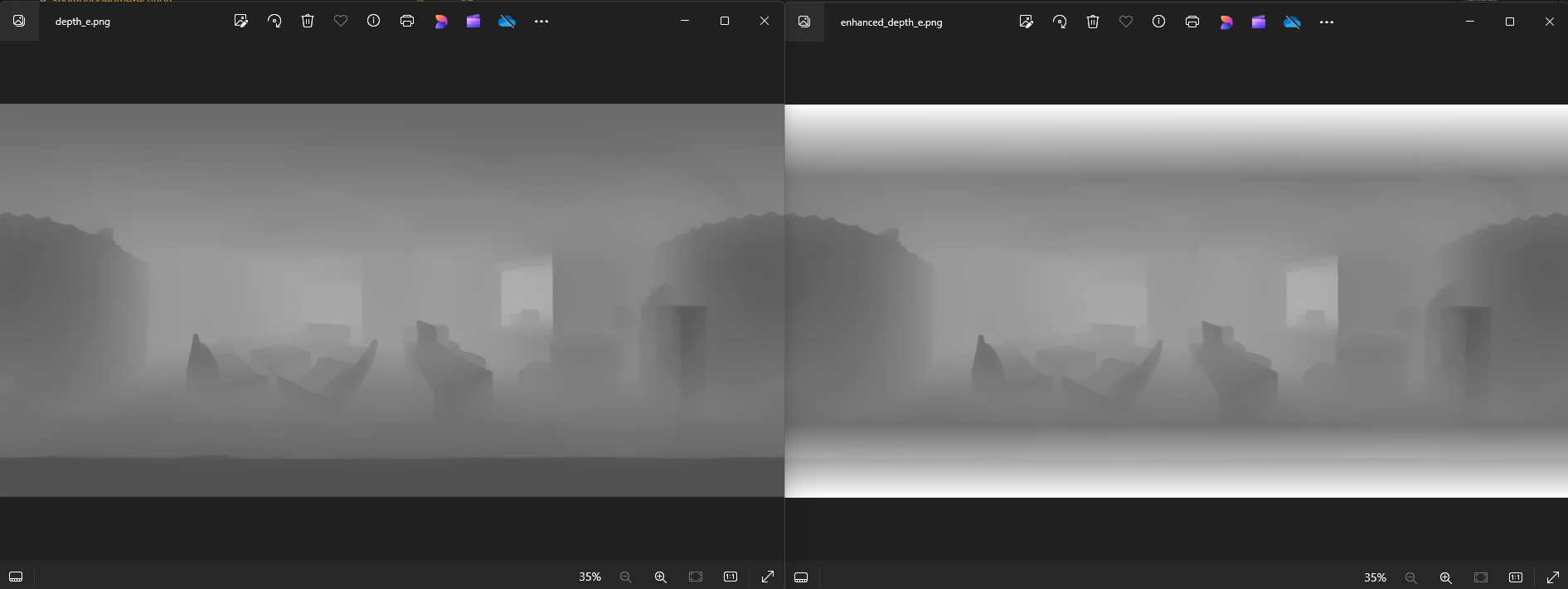


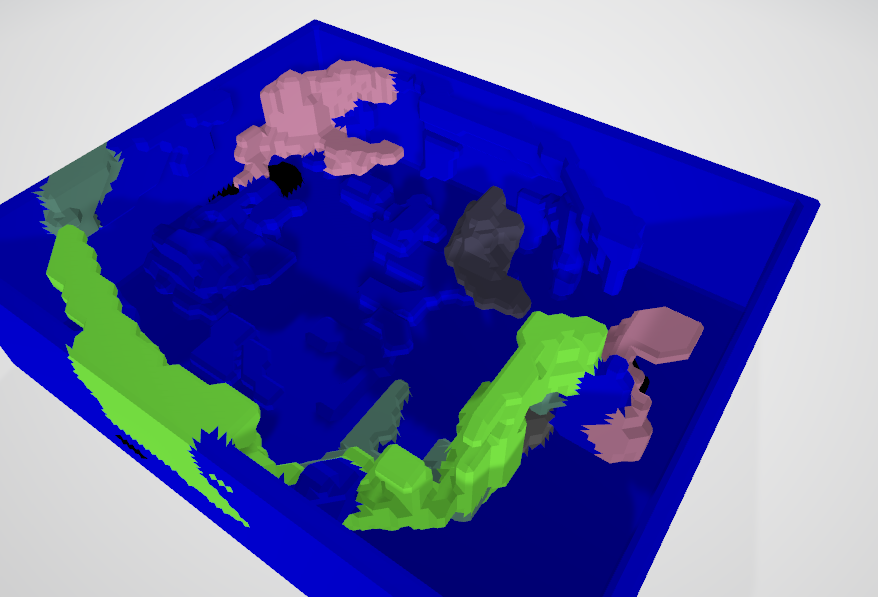
Monodepth v1.0 is the one with BoostingMonocularDepth.

  
it have different Image id as seen above two images, so replace all previous id with new one. (Need to refactor this so only define it in one file instead.

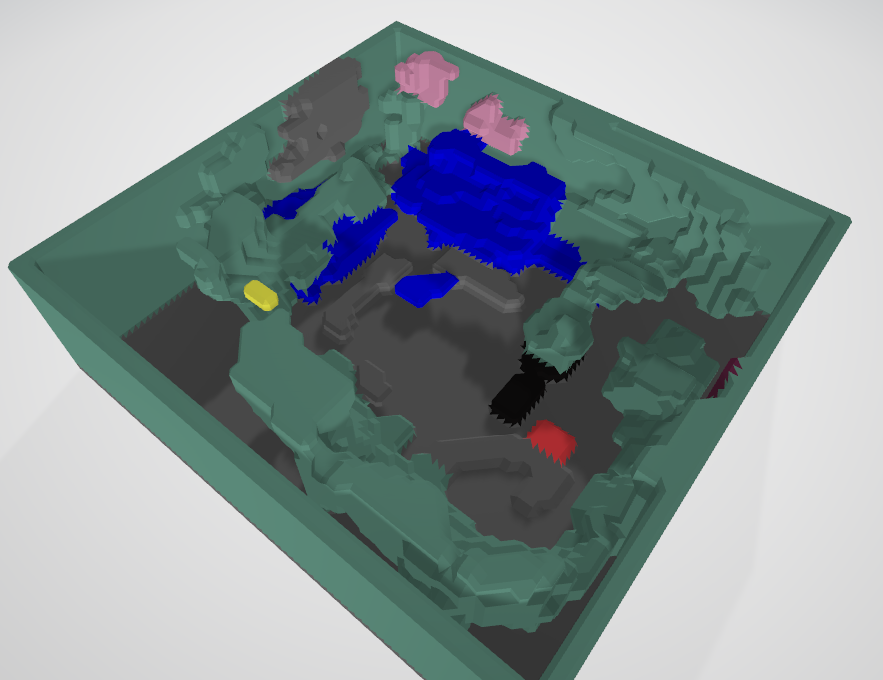


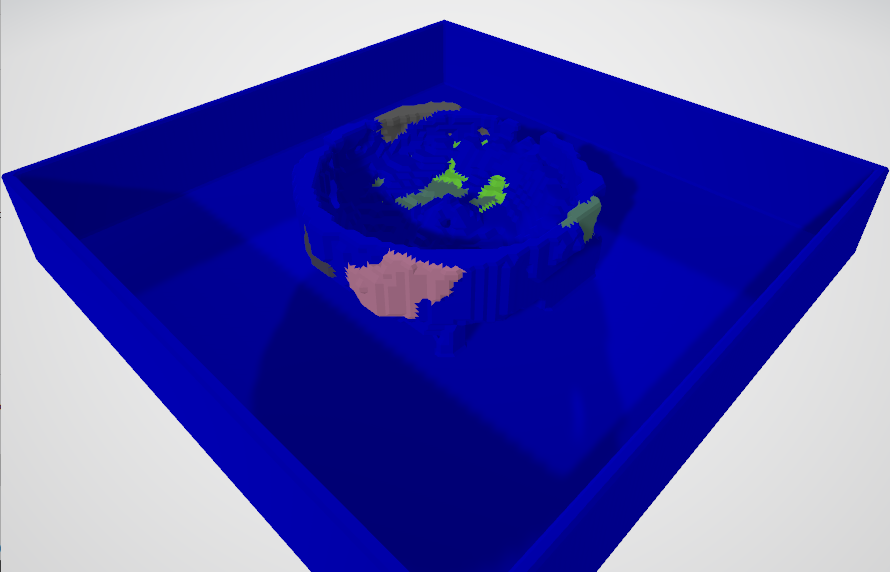
Enhance360.py at –baseline 2.264 vs new enhance360mono.py output that only gradient top and bottom to white, not much different but still can see that right is better in terms of depth details, the room structure might be a lot wrong now as no edge enhancement from rgb specifically for room infer360 made for edgenet and only use MonocularDepthBoosting for that edge enhancement etc. But need to check with mirror (UL scene). Now look at original enhance360.py default setting(baseline) against the new enhance360mono output. Left is sooo bad with lots of missing details and kinda inverted depth for sofa on the left side. But still, we need to see how well this depth map translated to meshes on edgenet, so lets see.

Ran on UL and can confirm MonocularDepthBoosting is still not as good as manual edge optimisation using rgb.png as in enhance360.py 

Left (original depth map), right is depth map output from enhance360mono which don’t have edge optimisation. As seen on mirror on the right, the reflection is still rendered on depth map instead of having only one plane depth.

UL generated mesh. Also causes these artifacts due to presumably unoptimised depth map as compared to enhance360.py, which means I probably need to make a mono version for most function in preproc.py instead.



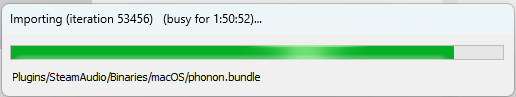
Not as bad for MR especially now we can see the sofa and table more clearly, but still not as ideal as stereo one or previous one in terms of clean mesh imo. Maybe I’ll just generate all scene and ask Dr. Hansung Kim which one he prefer and if I should work on it more etc. 

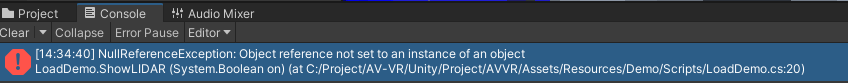
Bruh, I think the V2 have too white thus why give out such crazy room dimension (too large, tbh I should probably just normalise it to be lower, maybe have scaling factor to affect the output depth map as well.)

Rn, worst case I’ll just go back to normal enhance360.py with –baseline 2.264. ( still so bad esp in some scenes with stuff closer to camera… like UL, but idk, lets figure out monoenhance tomorrow.

# Tuesday, 16 July 2024

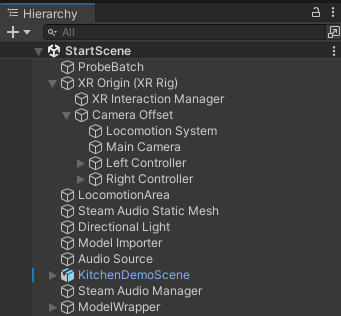
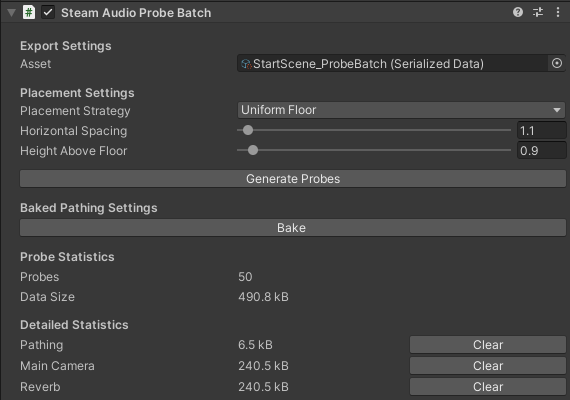
Now reconfiguring the StartScene to include Steam Audio Probe Batch and adding baked source and listener component to audio source and listener respectively. Also changed the behaviour from realtime to baked. This is per Mona’s recommendation. Apparently, the way to export audio is via external application such as Wavepad (recommended by Mona) but I think Audacity would suffice as that is what GDP’s group used. Just make sure to record in stereo mix/channel.

Its so joever

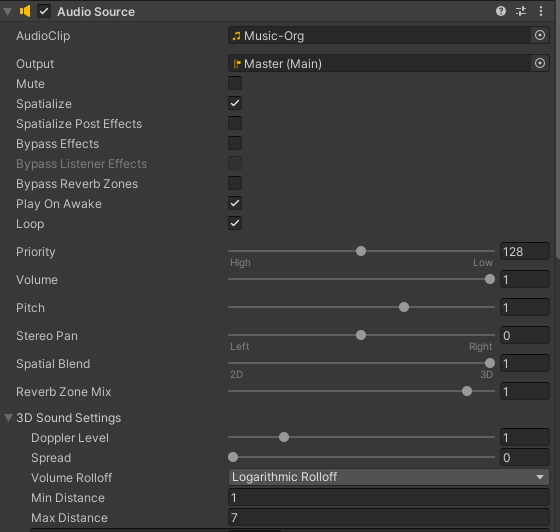
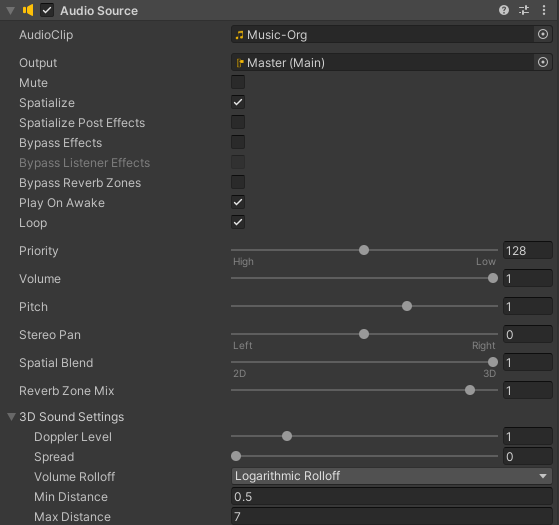
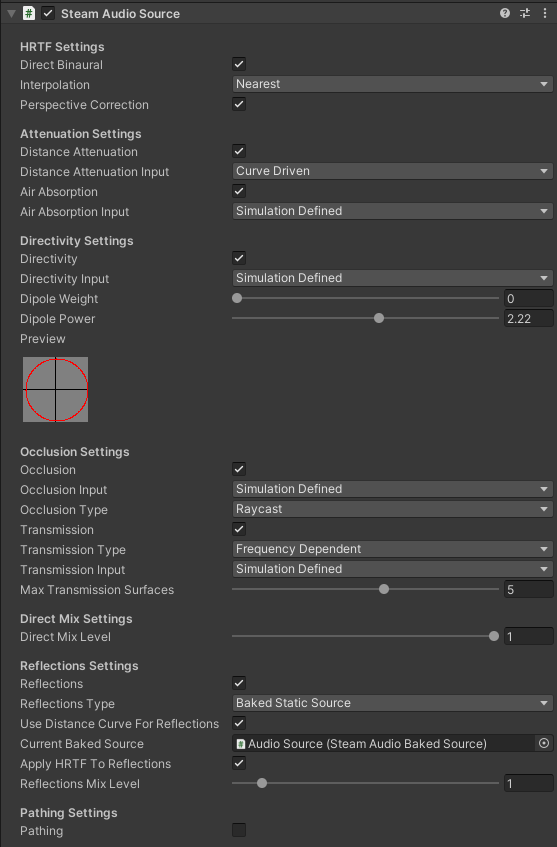
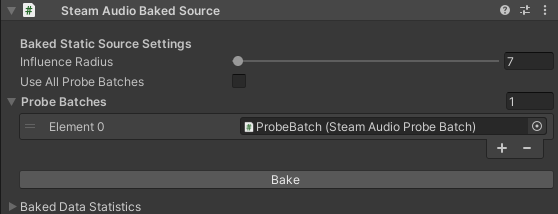
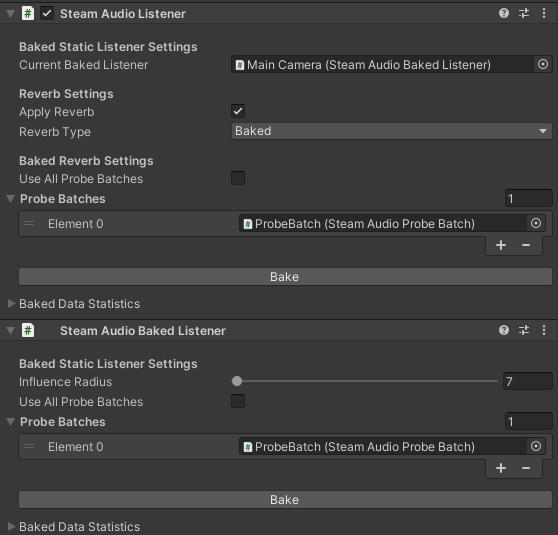


Need to fix this, either add checks to skip when not relevant

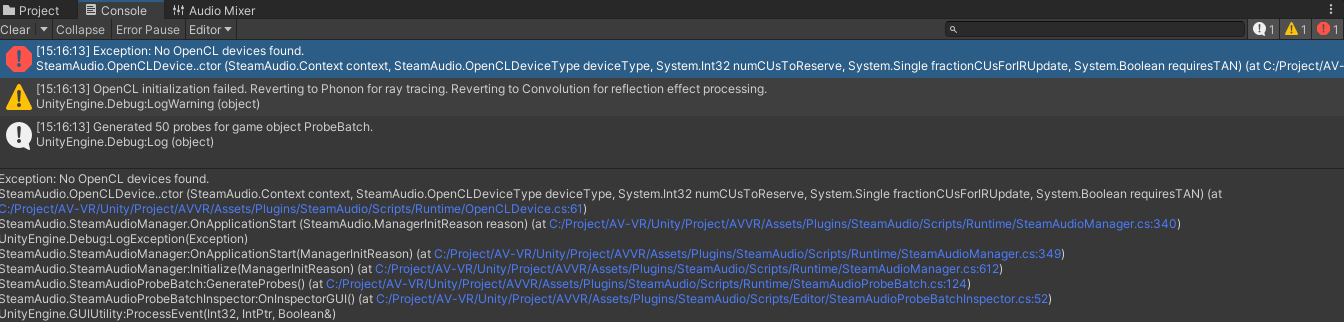
Ok added checks fixes and also for import scenery, added dynamic project dir instead of hardcoded.

Project hierarchy so far.

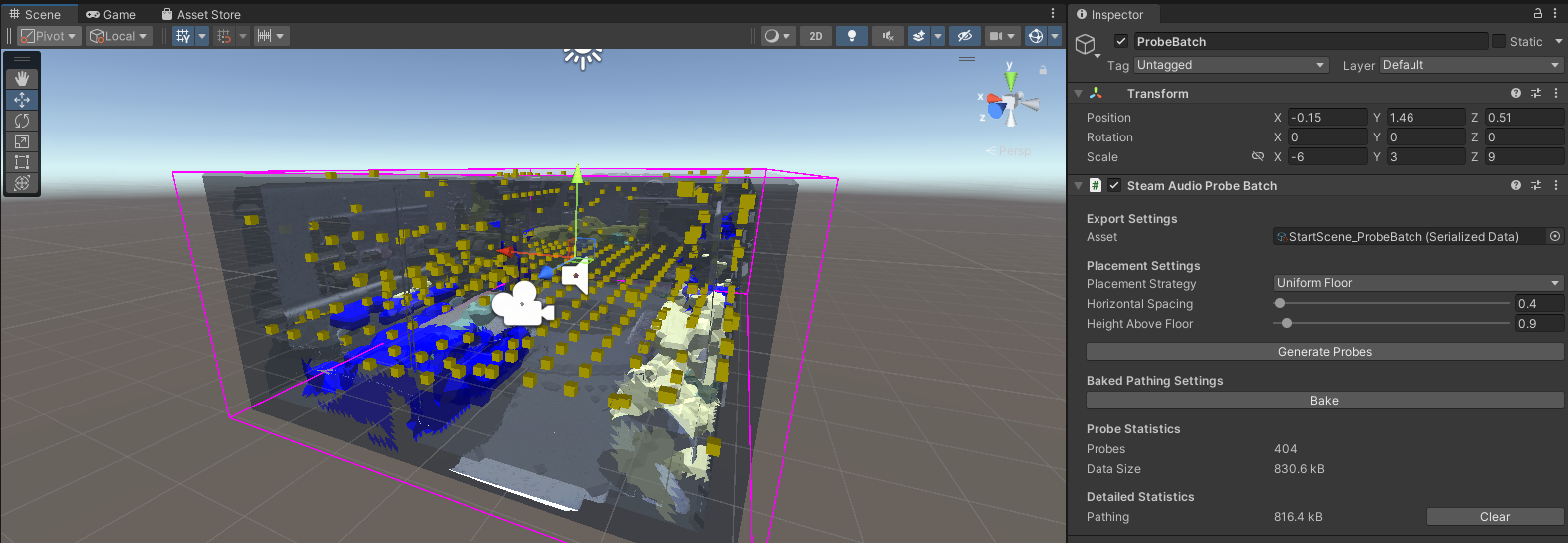
Probe Batch settings follow S3A equivalent scene (in this case, KT)

The same for all other settings except some are new so followed Mona’s suggestion. Apparently, based on my evaluation so far, Baked setting does sound clearer directivity and reverb wise compared to realtime, though that might just be because some other settings like Mix and rolloff also changes.

Remember to always export active scenes after every bake etc.

Weird bug, why is it looking at ‘AV-VR’ (old project directory) instead of ‘AVVR-Pipeline-Internship’ (new and current project directory). I’ll try reimport steam audio and see if this fixes it.

Ok, reimport hangs, so simply delete steam audio folder with its .met in plugin folder then reimport latest one fixes it.

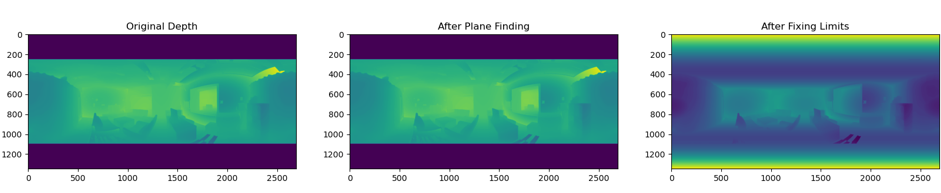


Note that for baked probe, it only have one level/layer thus why there’s height above floor for uniform floor setting, so using freecam mode is not realistic, thus why a player controller is needed with collision.

For some reason Radeon Rays mode give OpenCL error, also im unsure whether I should use TrueNext or not in reflection mode/type in Global Steam Audio setting.

Note: If baking take very little time especially on Listener, then there must be something wrong somewhere.  
There is weird occlusion+ distance attenuation bug I think which causes the sound to keep playing at normal/loud volume even at outrageous distance when leaving mesh at certain angle. Freecam related bug basically…

Gave up on SteamAudio and unity rn, I feel like its so buggy and hard to determine when its actually working or not, maybe I should have a log/debug or even ingame indicator to see if steam audio running/working as expected.

Now lets continue enhance360mono thingy, creates a visualiser script to see what each major step of enhance360.py did 

And from there, we can see that fix limits is the issue here, it remove the mirrored thing as its out of limit (far) but also remove near object (close to center), I assume this is because it thought it was part of floor. Maybe I can fix this by faking the closer object to have stronger depth or fake depth line. Or fix limit while taking into account edges(this most likely will bring back mirror problem tho).

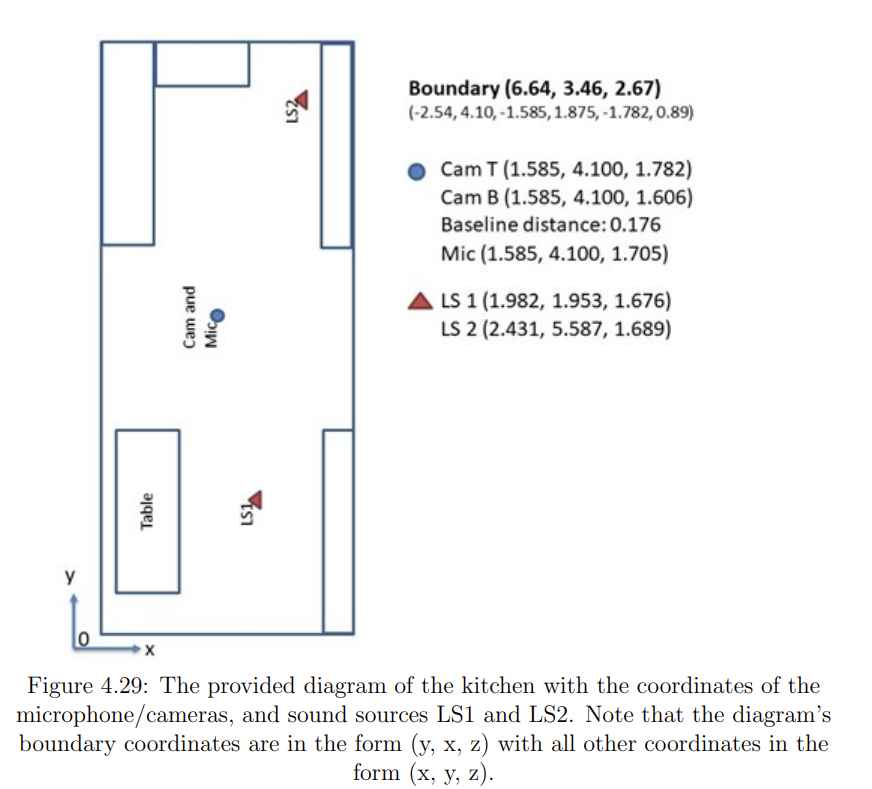
Looks promising now, fix\_limits\_mono from claude doing pretty well (the code version after I prompted patches of black/white) [Depth Map Enhancement for 360-Degree Imaging - Claude](https://claude.ai/chat/9fe50efd-9b07-4d45-95bf-39ad6a2db238)

Gonna sign off for today.

# Wednesday, 17 July 2024

Got MATLAB and python code for audio evaluation from Mona. So ig for now I should play around and get some results to compare.

Now there’s also another model with the best reproduction (stereo apparently).  
After looking back at GDP’s KT LS1/LS2 position images (note: where can I get these for other scenes? Or did they redraw it themsleve?) and old S3A unity project config in KT\_Eval



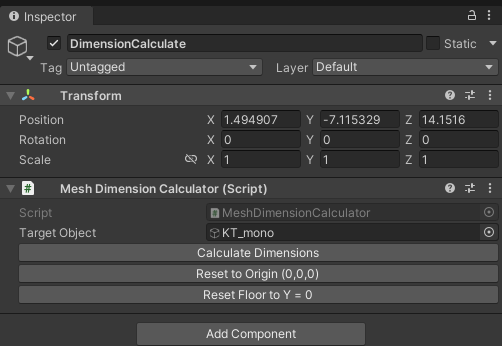
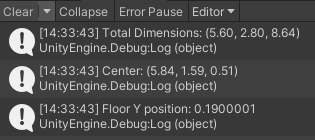
Im convinced that this is the correct position for cam/listener and 2 audio source. Another thing, because the unit is in meter, I should also just scale the boundary/scene size to follow so its easier to determine where each point should be (standardise it basically), and considering we have separate different scene saves for each thing (demo/eval/dimension comparison), might as well reset all to origin for setup at least.

**LAST TODO:**

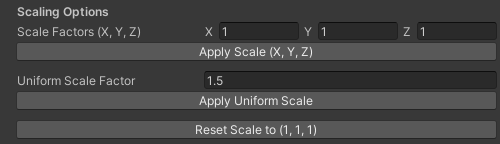
1. Figure out Steam Audio quirks and revamp Unity project
   1. Ask Mona for old unity project for reference, done and also for positions of probe/listener/source.
   2. Get MATLAB code for RT60 and EDT calculation. done
2. Continue monodepth optimisation otw, updated to use enhance360debug.py
3. Remove docker clutter
4. Remove LiDAR from LFS, and use separate cloud service to download that, or use MeshLab to optimize it.

**UPDATED TODO:**

1. Confirm steam audio setting and standardize/clean project directory to be less convoluted and organised
   1. Ask Mona for positions of probe/listener/source as pictured above for each scene.
   2. Confirm Steam Audio setting with Mona for each scene.
   3. Reorganise project directory
   4. Add lighting indicator for when steam audio is working
2. Add tools/script to streamline evaluation audio source/listener positioning
3. Add Rachel and Joao for Demo for more immersive VR demo, and probably have more audio cue.
4. Continue monodepth optimisation using enhance360debug.py
5. Remove docker clutter
6. Remove LiDAR from LFS, and use separate cloud service to download that, or use MeshLab to optimize it.

Created a helper scripts/gameobject to calculate dimension of model and recenter/refloor when needed. It’s pretty nice and can work with any imported via AVVR script as long as they have same hierarchy. This is important so we can center it to origin and get an idea of the ratio for positioning the probes/listener and audio source for evaluation.

For now, it shows on console logger which is a bit annoying imo cus it keeps stacking up but it work for now, better than nothing.

Also added scaling options so its easier to scale and check dimension in one menu!

Hmm I just realised the problem, idk what theboundary here means, are its using 0,0,0 as origin even? The diagram showing position I mean. Ah just realised the 0,0,0 is at the edge of kitchen bruh.

Added button options to reposition and rotate so that the model edge at 0,0,0 and extrude towards +ve x and +ve z axis. Ok this gonna be harder to code/prompt than expected. Idea so far is first to rotate to correct orientation THEN bring the corner to 0,0,0.

Nice, it works perfectly. Only problem is the dimension of monodepth is so wrong, so I need to find ratio to scale them to correct dimension first, or else I cant really use the coords provided unless I scale them first, which is more annoying so might as well find ratio first ( I need to find it anyway)

Also, changing the coord to be x,z,y because that’s how it defined in the diagram. OK NVM, this is confusing, and make it harder, lets just use normal XYZ or unity, and map the diagram to proper XYZ lmao.

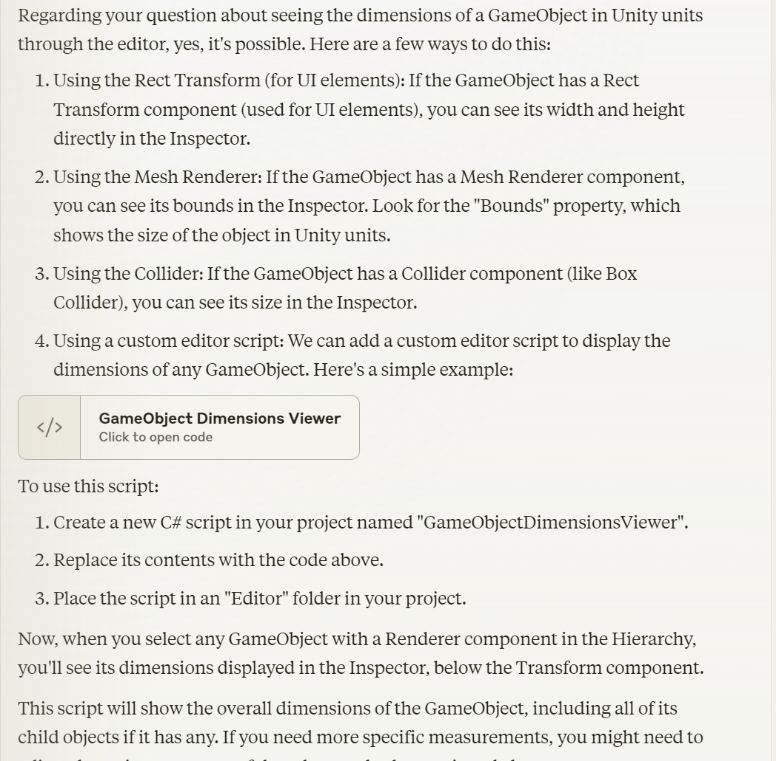
Dimension (x, y, z) of scene meshes from original enhance360.py and 360monodepth to calculate scale ratio needed as follows:

1. KT – (4.16, 2.08, 5.44)
2. MR
3. UL
4. ST
5. LR

|  |  |  |  |
| --- | --- | --- | --- |
| Scene and ratio (x, y, z) | Cam/Listener coords | Audio 1 coords |  |
| KT (0.83, 1.28, 1.22) |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



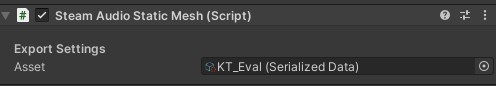
From <https://doi.org/10.1007/s10055-021-00594-3>, in Z, X, Y ( Length, Width, Height)

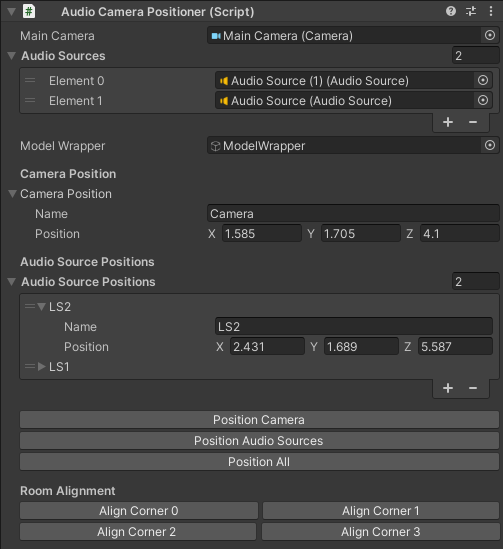
Unrelated but just realised I could just use a simpler way to get dimension of the modelwrapper, either instantiate it with trigger/disabled collider or put transparent mesh renderer over it.

This require me to modify some part of ImportScenery.cs though. Ok nvm, idk if instantiating collider or renderer through script would even inherit the whole meshes dimension, and current script works well anyway so oh well.

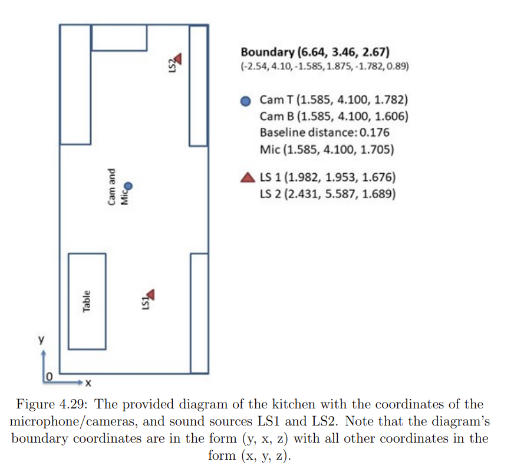
Did it anyway so can double check/confirm. And can confirm it is correct calculation for both

Found a bug with

Remember to set this to None and create new one for new scene.

YAHOOO FIXED AND IT WORKS PERFECTLY (AudioCameraPositioner editor menu helper script)

Having all corner options make it easy to rotate no matter how the diagram is made as long as it follows positive XZ quadrant and start at 0,0,0 on corner of mesh!

For ref, KT diagram given:

Done for today ngl.