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

Done, Important

# Monday, 08 July 2024

New week, new day, new season (its not summer m8, its raining whole day whole week wth)

TODO unfinished from last week:

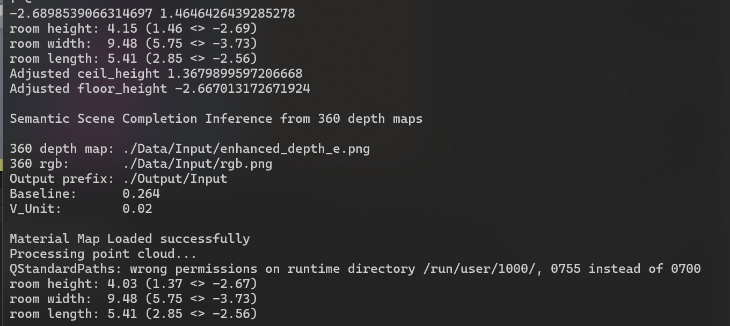
1. Make monodepth depth map ‘stronger’ to work with enhance360 better.
2. Remove old docker containers automatically to remove docker clutter
3. If provided tools/methods to continue with audio test/eval, do that, if not,
4. Maybe make the project/pipeline more easy to reproduce/run by having a .txt for custom project dir etc so no need to replace all manually on vscode etc
5. And also fix git repo to make it pullable easily (submodule not pulled, and LFS for LiDaR)

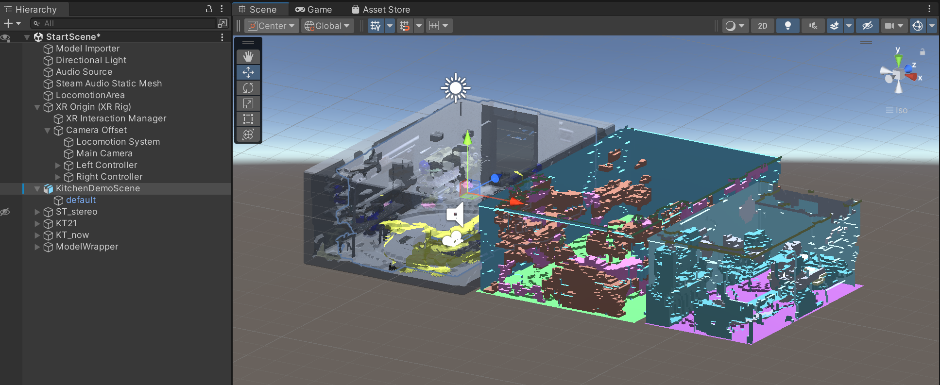
Another problem? For some reason, I feel like the output of material assignment is not consistent for some scene (UL).

Anyway, the idea to fix no. 1 is to add pre enhancement step to have the depth map ‘stronger’ or more pronounced, but need to check if resulting room dimension still within same range as ground truth or stereo matched depth map. Also, maybe add a pseudo ‘error’ or outline to mimic stereo depth map as well? Post process is not possible as the depth data is already lost at that point and cant be recovered.

Ground truth, stereo depth map and monodepth map room dimension table (WxLxH), first two taken from [Kim21 on Table 2](https://link.springer.com/article/10.1007/s10055-021-00594-3)

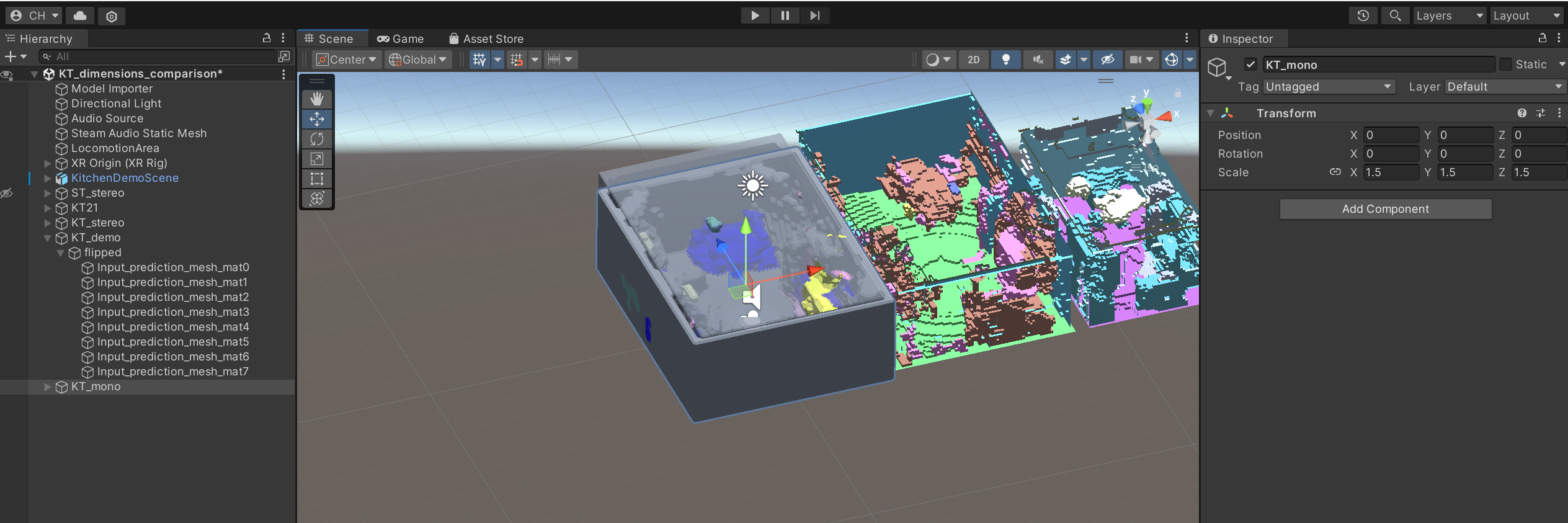
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scene | Ground Truth | Stereo (Kim21) | Monodepth | Monodepth (enhanced) |
| MR | 5.61 x 4.28 x 2.33 | 5.54 × 4.24 × 2.40 |  |  |
| KT | 6.64 x 3.46 x 2.67 | 6.42 × 3.52 × 2.68 |  |  |
| LR | 5.64 × 5.05 × 2.90 | 5.88 × 5.02 × 2.78 |  |  |
| ST | 17.08 × 14.55 × 6.50 | 17.54 × 15.46 × 5.56 |  |  |
| UL | 5.57 × 5.20 × 2.91 | 5.52 × 5.22 × 3.00 |  |  |

OK, I have no clue how to get the dimension… Using the edgenet360 printed dimension on enhance360 and infer360 is very different from ground truth and stereo (Kim21).

Dimension for KT using stereo depth map. Very different, however it fits the given LiDaR scan of KT the most… What?

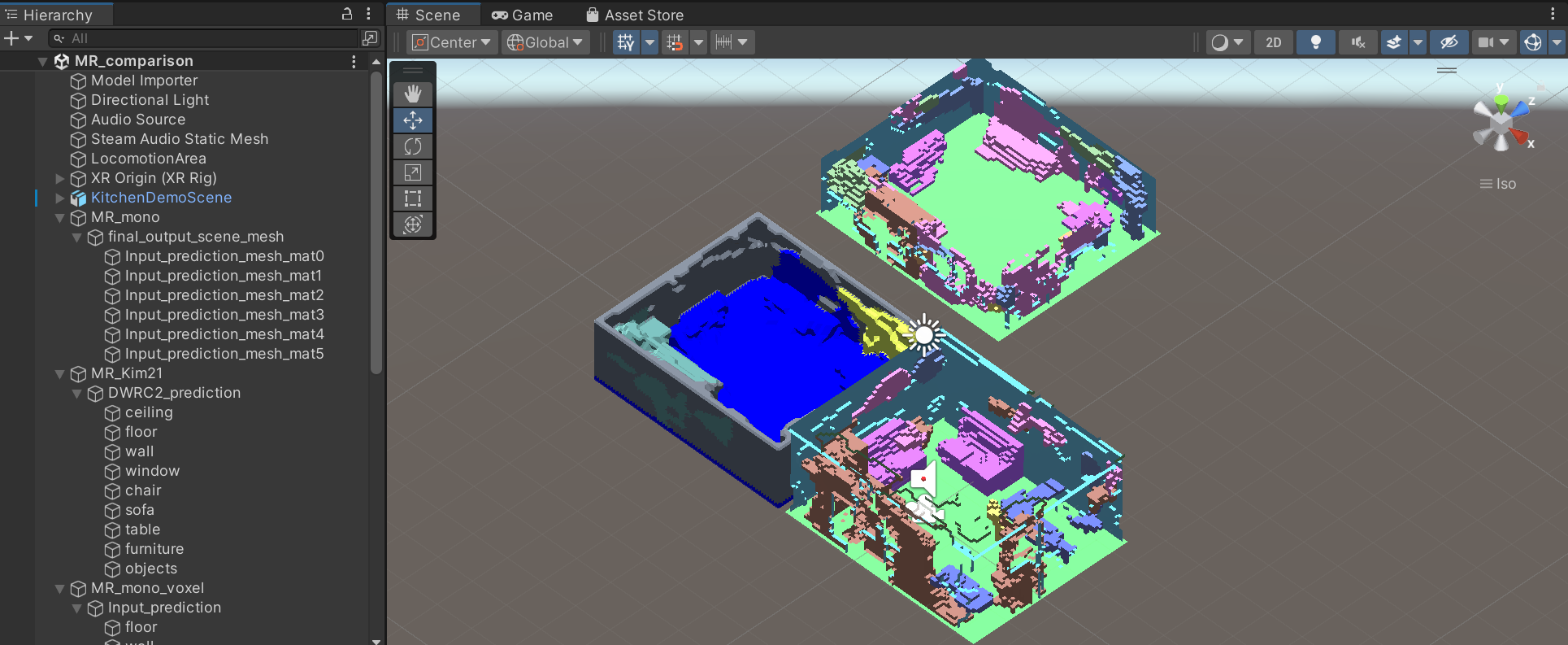
LiDaR (demo), KT\_stereo and KT from [Kim21 data](http://3dkim.com/research/VR/index.html?ckattempt=1) (which should be the best)

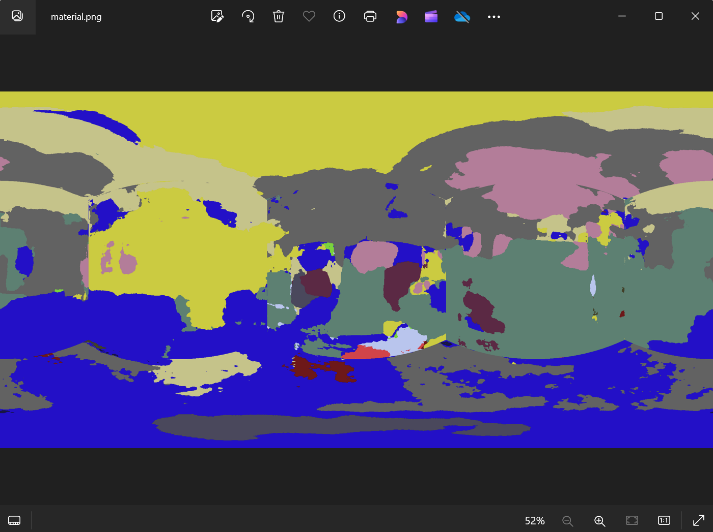
KT\_stereo is generated using enhanceori.bat in enhancetest folder in scripts.

Fwiw, scaling KT to 1.5 on all axes gives a pretty good approximation to Demo and Stereo.

Need to ask Dr. Hansung Kim about this for more clarification. For now, considering I can probably just scale it manually later to fit the dimension (the original im getting rn is so broken anyway, atleast compared to data on table), I’ll just try experiment more with this preprocessing steps and see if result better visually (less/not empty in middle).

UL Kim21 cant be imported using Import Scene or import .obj on Unity, and when opened in 3d viewer, it looks broken and no material (broken normal map? Need to check on blender).

Next comparison, shifted MR mono, overall not bad except height is tad bit shorter:  


Note that due to the way pipeline implementation is done, the details regarding the object details like sofa does not exist in final scene mesh .obj but it does exist in voxel mesh… This is a bad oversight imo because it make it harder to see which object is which and orient mesh properly without it for evaluation.

Especially now that material recognition have issue regarding the missing global context when separated to cubemap projection as seens on picture below and related cutoff point that the table is misidentified as fabric due to the cubemap projection limitation as seen above.

Absolutely horrible ending, got some weird filepath error on main GUI.py pipeline wth.. im tired so gonna debug tomorrow instead…

# Tuesday, 09 July 2024

Fixed by reverting the combined.bat changes on monodepth copy etc, but now I cant use shifting…. Got the cannot perform cyclic copy error… The mistake before is that I didn’t check/test the original functionality after testing that shifting works (cus now non shifting doesn’t).

Refactoring the whole combined.bat to be more modular with hardcoded file/folder directory being defined on top.

Having lots of buggsss….

3rd meeting w/ Dr Hansung Kim, points:

* Getting correct dimension from monodepth not possible as no reference/baseline like stereo, best rn is to get correct ratio scale and find scaler parameter.
* IMPORTANT! Figure out why steam audio on GDP project sounds weird/not correct! (maybe redo steam audio from beginning even!), check api/manual to make sure correct setting. Biggest problem is not enough surround sound effect/directivity/HRTF(?)
* Fix/optimise mesh more if possible
* Audio eval code (matlab) is w/ Mona

So according to this, previous TODO:

1. Make monodepth depth map ‘stronger’ to work with enhance360 better. **Tried!**
2. Remove old docker containers automatically to remove docker clutter **NOT YET**
3. If provided tools/methods to continue with audio test/eval, do that, if not, **NOT YET**
4. Maybe make the project/pipeline more easy to reproduce/run by having a .txt for custom project dir etc so no need to replace all manually on vscode etc **OTW**
5. And also fix git repo to make it pullable easily (submodule not pulled, and LFS for LiDaR) **NOT YET**

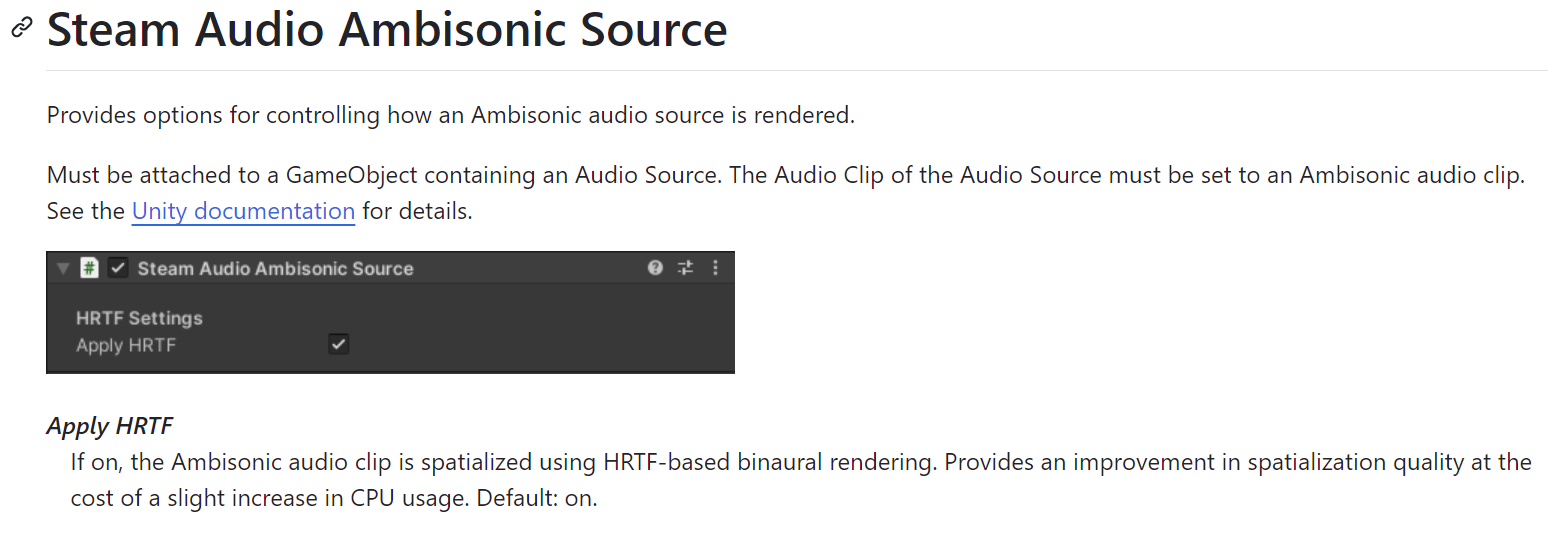
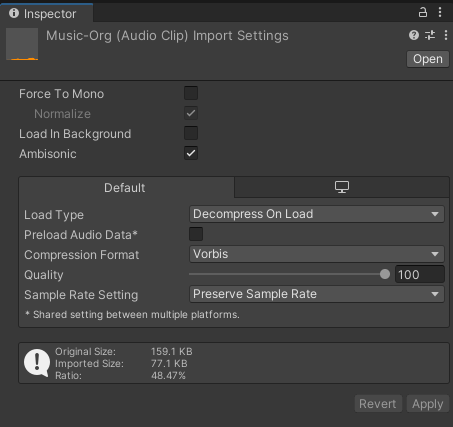
Combining them, with highest priority first, **UPDATE TODO:**

1. Refactor pipeline to be more modular and fix bugs done and fixed (prob)
2. Figure out Steam Audio quirks and revamp Unity project
3. Continue monodepth optimisation ( fake stereo depth using mono?)
4. Remove docker clutter
5. Fix git repo (submodules and LFS problem)

Fixed refactor, problem is .jpg and .png lmao. Anyway, it should be more modular and expandable now.

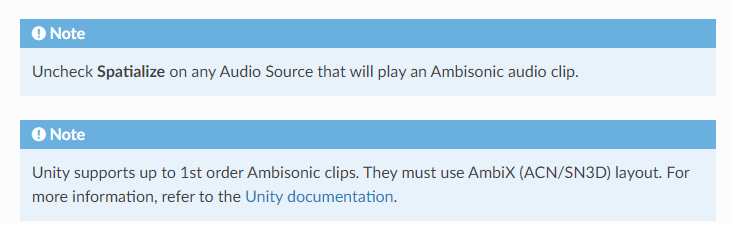
NVM, shift option still so buggy (not working with mat recog). NVM, its broken all!

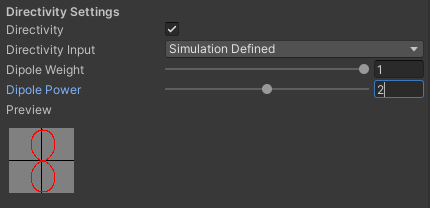
Steam audio, maybe this is the culprit? Tried adding it.



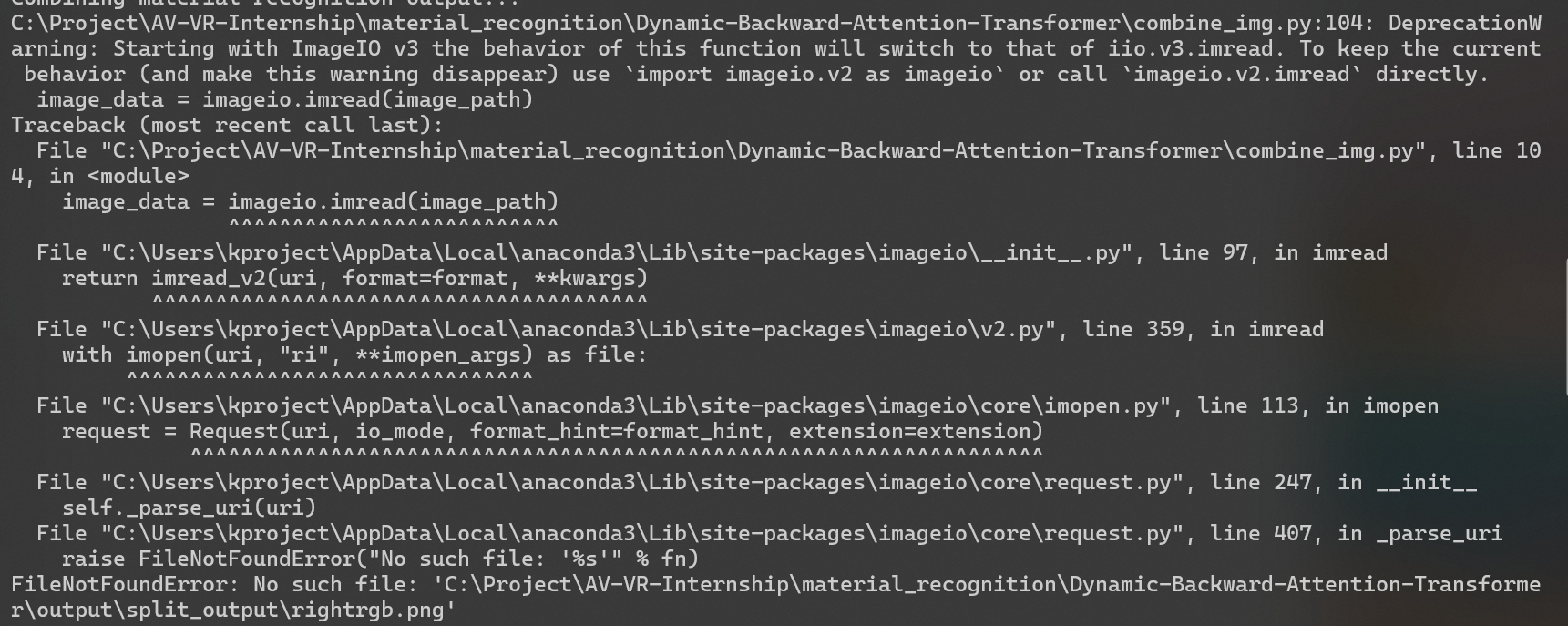
Also ticked ambisonics on the audio clip source itself. Honestly idk if im tripping but there is a difference, albeit very subtle. Maybe should try in ST instead of KT. IDK MAN, after more testing (turn off/on) even for speech, it feels like no difference, heck, I feel like default (GDP’s) setting have enough/better directivity already.

Ah, refer image below, maybe that’s why. NVM, ambisonics files have different format, this is mp3 which most likely not ambisonics format.

For some reason the steam audio github page is down, so Im using wayback machine to access the docs

This is most likely the answer, the dipole of the audio source would make it simulate real speaker more instead of omnidirectional speaker.

That aside, lets go back to the broken pipeline YAY. Reminder to delete intermediary output to make sure the pipeline is really working!



WTH??? I deleted the rgb split output but wdym it don’t exist no more oh uh. Ok fixed it finally, all this time it might just be because using wrong ‘\’ instead of ‘/’ for directory and files… wth. Why does this matter so much smh. The problem is that the rgbsplit of material recog output don’t seem to be working. Changing the backslash form fixes it…. Idk man.

KT w/out ceiling and not shifted works (using shifted\_t alr) !!

Now testing shifted MR w/ ceiling. It works!!!

Gonna commit and push tomorrow instead.

Technically not finish refactoring yet, there still exist directory hardcoded in scripts such as split\_img.py etc.. need to find all instance of hard coded and refactor/optimise those as well! Also need to add function to remove temp/intermediary output after copying to make sure rerunning don’t depends on old/existing output but generating new one. This is oversight! At least for now, Phase 1 refactor complete!

# Wednesday, 10 July 2024

**Updated TODO:**

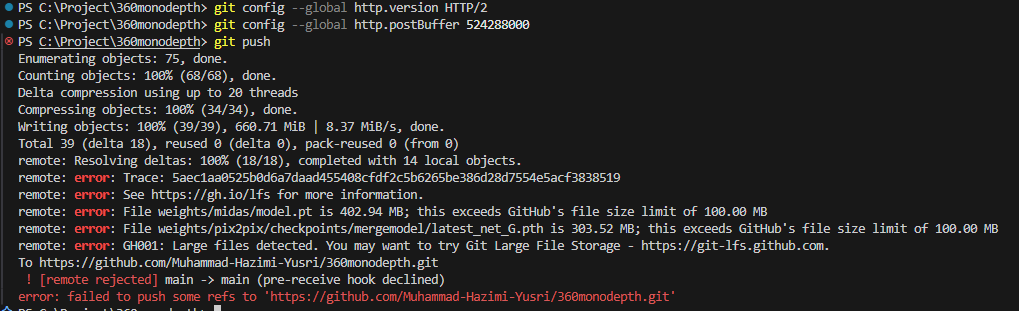
1. Figure out Steam Audio quirks and revamp Unity project
2. Continue monodepth optimisation ( fake stereo depth using mono?)
3. Remove docker clutter
4. Fix git repo (submodules and LFS problem) done
5. Refactor phase 2 (other scripts and cleaning up) done

Doing 4. rn, added LFS but remote branch for some reason got separated. Tried to fix but created more mess. Gave up, will just create fresh new repo instead.

Dealing with submodules, so to make sure submodules changes are updated according to what the pipeline needs, I forked all related submodules first (and for simplicity, keep them public). For some reason, my Vscode git pushes have authorisation problem, so I just push using Github Desktop instead.

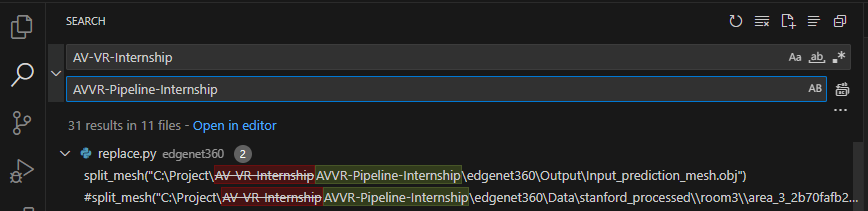
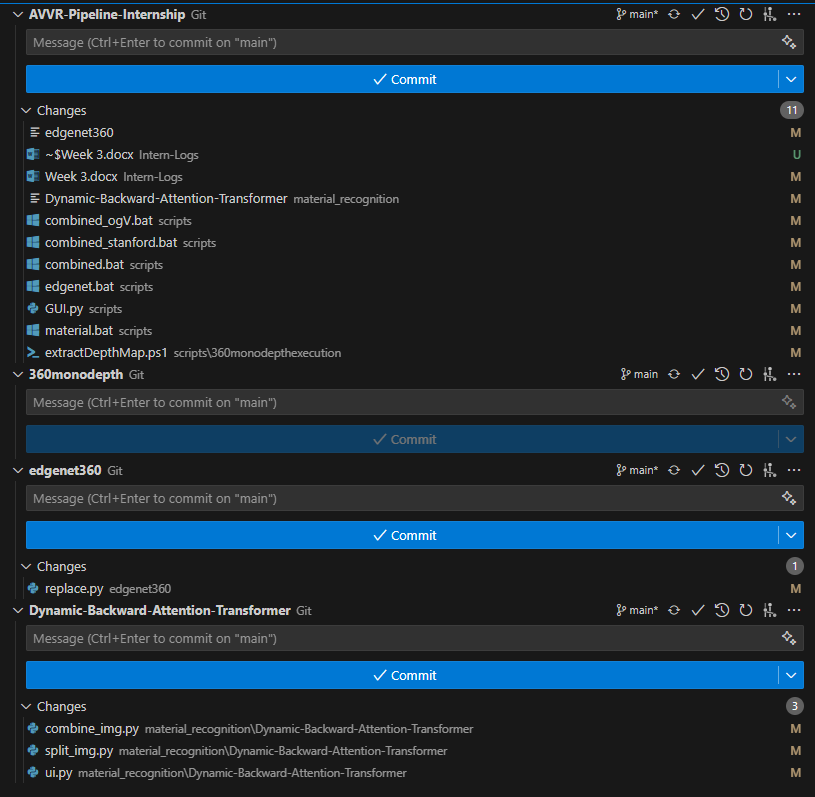


Getting this error, but figured out I am committing the weights (not supposed too! Read the instructions on ReadMe.md github repo page…)



Also, no need to fork BoostingMonocularDepth cus apparently GDP doesn’t use it, but maybe I should add it anyway to submodule.. Nice now just run

git submodule add https://github.com/Muhammad-Hazimi-Yusri/360monodepth.git 360monodepth

to add the forked submodule properly. Added all 3 forked submodule properly now! And gitignore also for each module weights/data etc. Now just need to replace all instance of AV-VR to correct dir.

for Phase 2 refactor, this needs to be

settled so only 1 file need to define this

main dir!!! Here I need to commit 3

different times!

Will test script functionality tomrw!

# Thursday, 11 July 2024

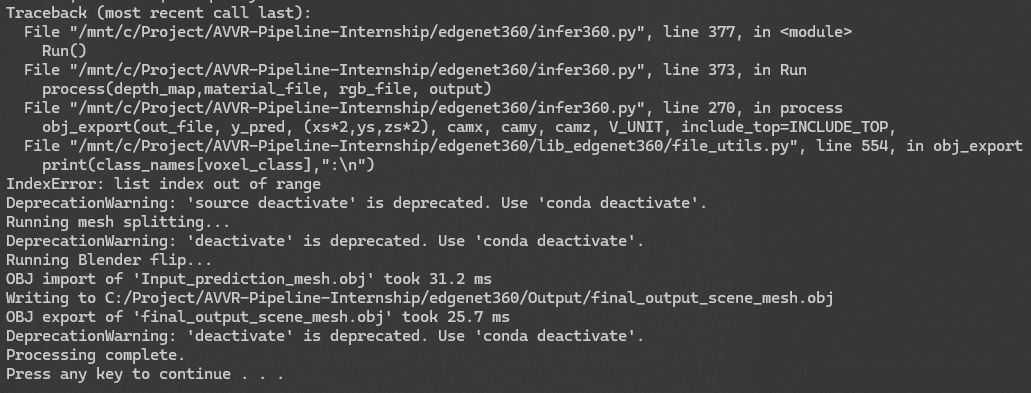
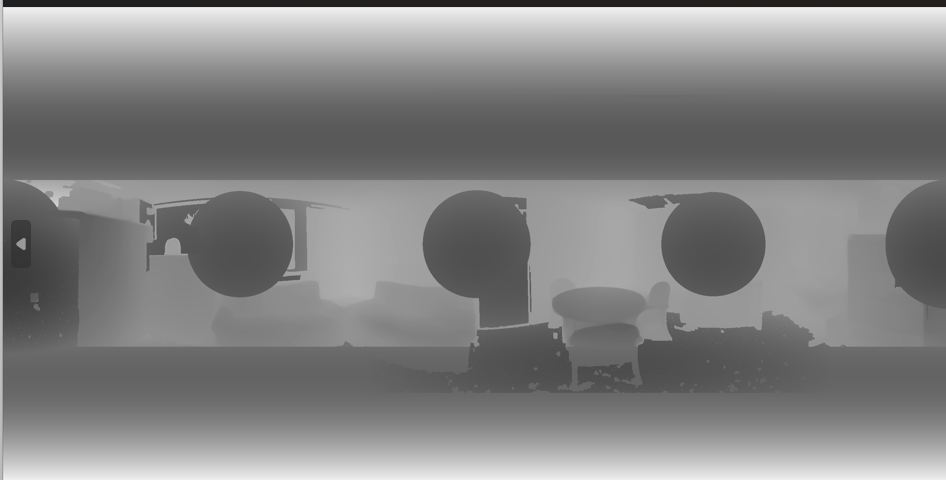
Now going to test if GUI.py breaks or not.

Confirmed that it works! The unity project also works! Now lets get back to   
**TODO:**

1. Figure out Steam Audio quirks and revamp Unity project
2. Continue monodepth optimisation ( fake stereo depth using mono?)
3. Remove docker clutter

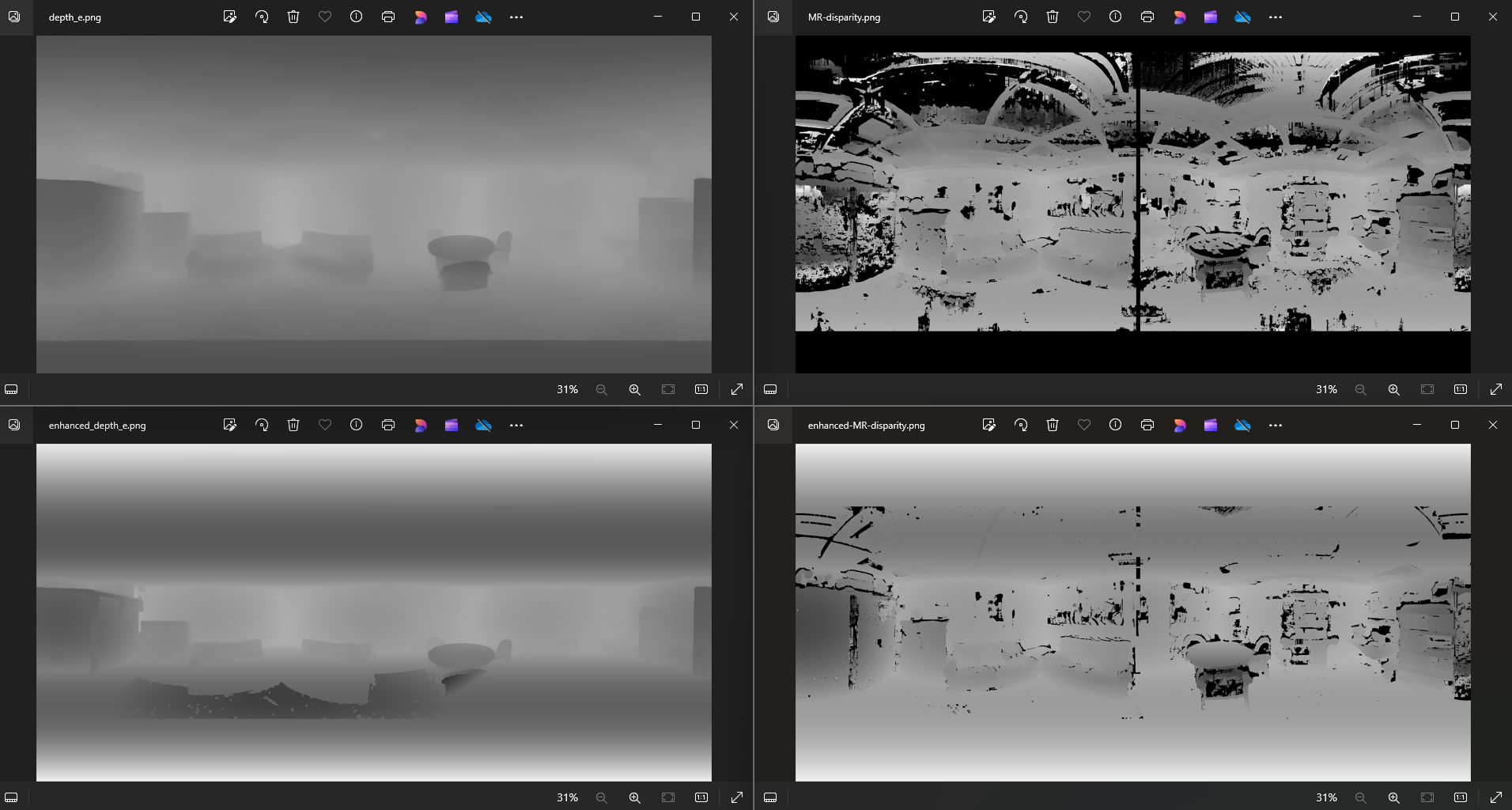
Ok, tbh for no.1, idk what I really should be doing… Maybe lets just make fresh Steam Audio project comparison with non Steam Audio etc.

For no. 3, as the exact dimension reproduction is not important for monodepth (no reference/baseline issue) to get exact dimension, ie the rough dimension is good enough as long as ratio of WxLxH is correct. This means I can go ham on pre enhance of monodepth to make the depthmap ‘stronger’ or fake stereo depth match artifacts.

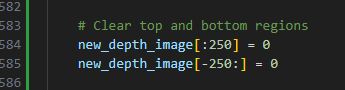
Trying not too modify too much or add convoluted steps like preprocess etc, I looked into the enhance360.py arguments and code to see if there’s simpler way to modify it to fit out monodepth image. Increasing baseline value seems to do the trick. Baseline is the distance between the stereo camera. In our case, increasing it means we want the enhance to not remove too much details and depth on our already smooth and ‘weak’ depth image (I think).

At –baseline 5.264, I got the error above, which I assume due to the following interesting depth map on the left lol.

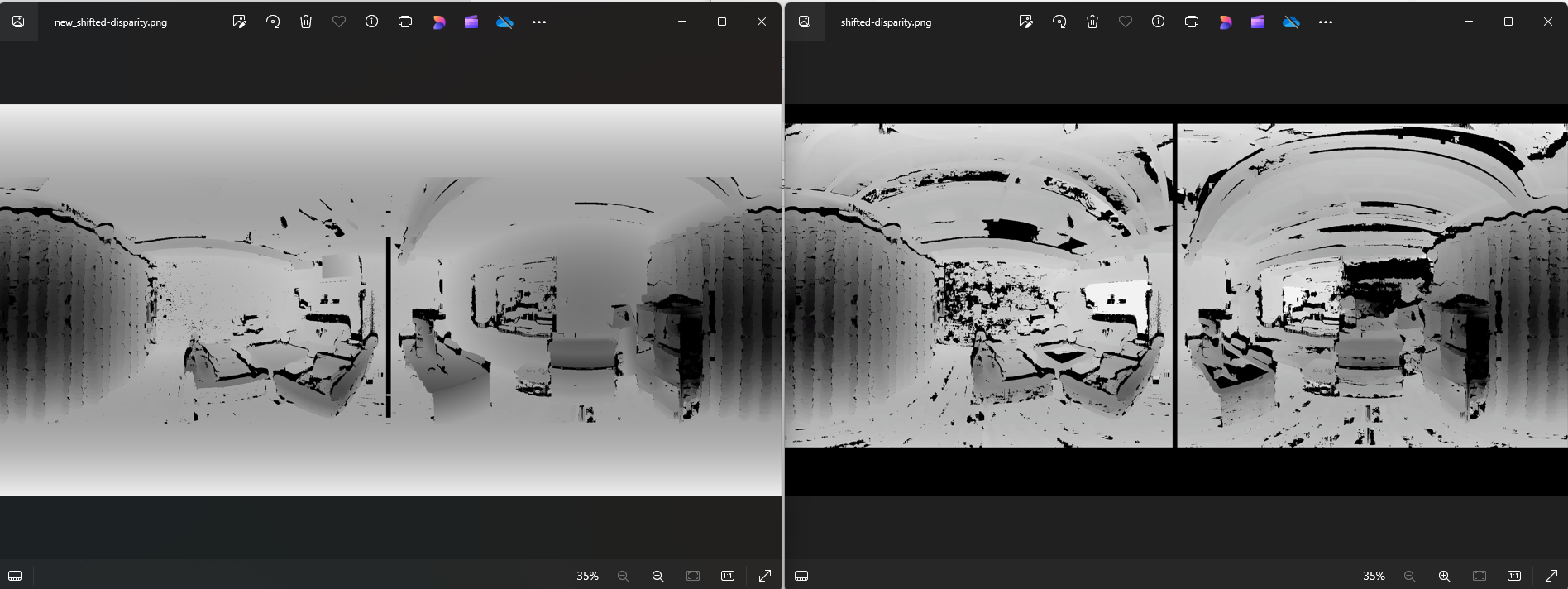
On audio side of thing, im following this lesson series [2. Sampling Theorem - Digital Audio Fundamentals - YouTube](https://www.youtube.com/watch?v=vrXGaFV1AmE&list=PLbqhA-NKGP6B6V_AiS-jbvSzdd7nbwwCw&index=2) Going to download Audacity and observe the recorded waveform from previous paper and current later.



Coming back to the enhancing step, I’ve realised that maybe I should comment all except ceiling/floor modification and maybe plane cus the enhancer seems to invert(?) or put gradient on objects it shouldn’t.

Most important part to remove top and bottom, its so simple duh. But now I kinda want to keep using the enhance360 as it have other useful output like for mirrors etc.

OK I think lets just ignore mirrors for now and do bare minimum preproc (no enhance360.py) until we figure out how to make it work better. Also lets test on UL to see which actually fixes the mirror

also just realises for our case, the mirror fix is not intentional, just lucky byproduct of edge detection enhancement thing.

I gave up for now, the commented both find\_planes and just add the clear top and bottom seems doing pretty good but still some depth lost… need to figure out where and why….

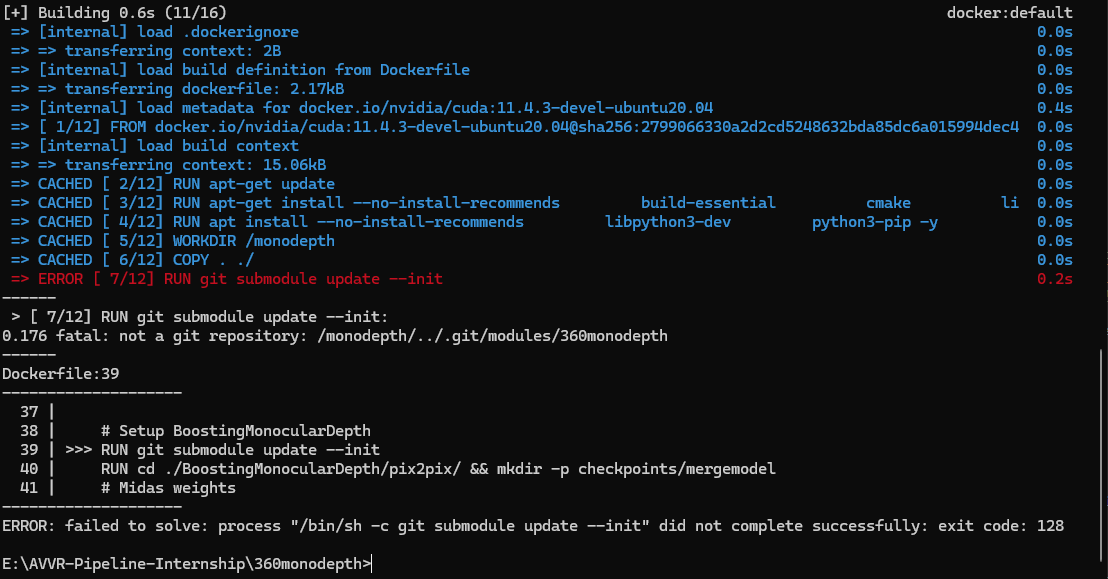
# Friday, 12 July 2024

WFH today, a bit more productive cus github pull now work with better/cleaner repo.

Unity project works but have some weird .meta deleted when ran, and sometimes the prefab not initialised properly (XR Controller) thus need manual reassignment.

A big issue with LFS, I’m already over the bandwidth limit of 1GB even though only pulled once… Maybe it always pull the .obj file everytime I pull? This mean git LFS wont work. Need to find better solution. Due to the import AVVR scene being related to the big kitchen lidar .obj, that means I need to do check for it exist first if not then no need to use to prevent compilation error, and use different storage provider for big files instead of git LFS (more convenient but too small bandwidth and storage rn).

Next, I should put some basic manual/todo checklist each time I import to new PC/system so that I install all prereq for running properly (wsl/docker and python env)

Lets make new Manual.docx for that ngl. Nvm maybe not, lets just revise existing one.  
  
Regarding enhance360.py, I’ve realised that boostingMonocularDepth on 360monodepth actually do some context/edge optimisation but GDP’s code commented/excluded the submodule implementation hmm, assuming due to the Dockerfile not working as its now a submodule instead of git repo. I feel like if I fix this, I can make the depth image better without needing enhance360.py (for mirrors and stronger/accurate depth) and just need to gradient top and bottom instead. 

**UPDATED TODO:**

1. Figure out Steam Audio quirks and revamp Unity project
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