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

Done, Important

# Monday, 01 July 2024

**360monodepth** error due to missed to change some instances of hard-coded directory reference (‘AV-VR’ to ‘AV-VR-Internship).Fixed now but This needs to change if want publish/easier use. Maybe just set the project dir etc on one .txt file and all scripts reference to that!

**Edgenet360** errors!

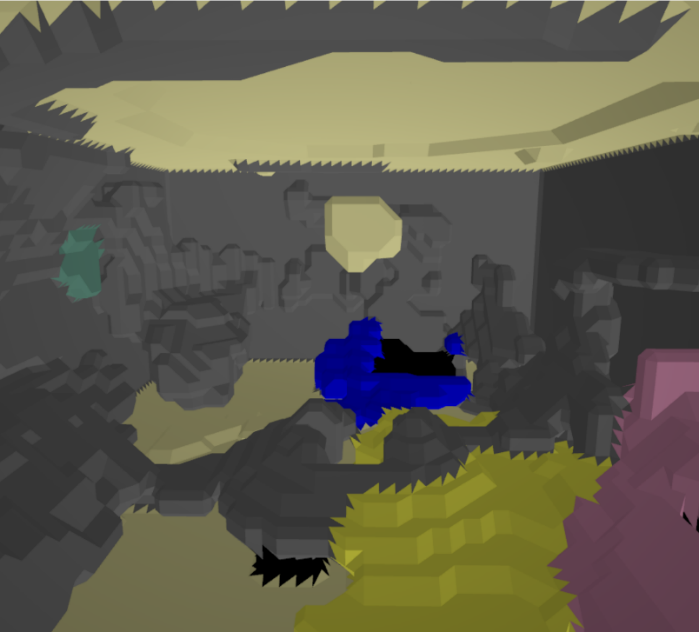
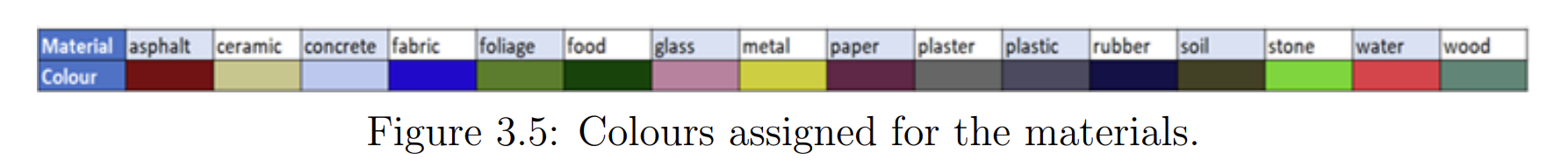
Missing miniconda error as there is no miniconda in WSL installed, thus I changed it to anaconda as that was installed, also followed the [github installation guide](https://github.com/SimonLisowski/Mesh-Generation) (albeit some links outdated, so I googled for some imports). Need to have version number as well so no dependencies/outdated issues… Original [Edgenet360 gitlab repo](https://gitlab.com/UnBVision/edgenet360) seems a bit more comprehensive

Value Error due to outdated TensorFlow implementation; Fixed by using .legacy import (Thanks to Mona for suggestion for easy fix).

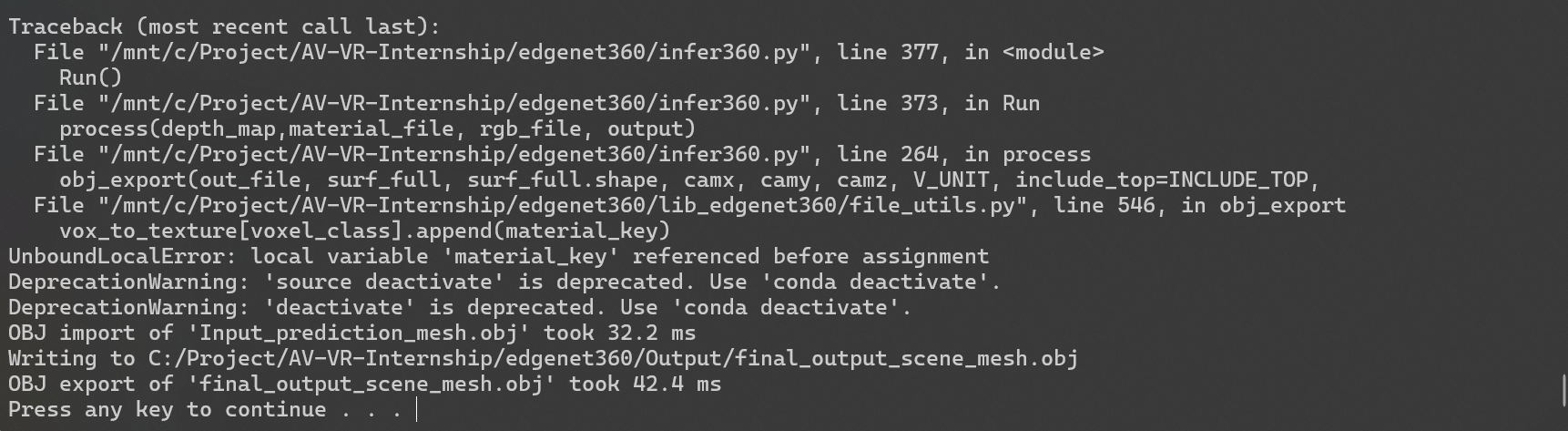
Index Error during smoothing part in postprocessing.py; Fixed by adding checks.

All other scenes generated mesh successfully at this point!

Need to check if smooth fix vs unsmooth (just add ‘--smoothing n’ as parameter in combined.bat line 46) have difference or not.

Not smoothed vs smoothed (visual comparison):  
Honestly not much difference meshes wise visually, but interestingly enough the material assignment for ceiling is different (Plaster on unsmoothed and Ceramic on smoothed). Why? Fluke? Need more testing later. The size is also different so it should do something at least. 

New error but only for Meeting Room! UnboundLocalError:



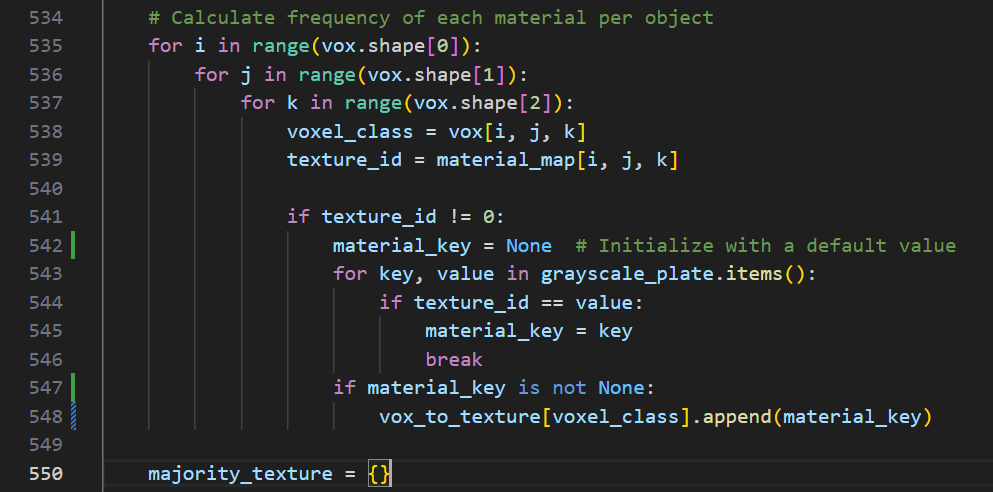
Is it because missing shifted\_disparity.png?

Need to check! NVM, most likely not because of that.  
It is probably due to

“The error indicates that material\_key is being referenced before it is assigned within the loop that processes the voxels and texture IDs. This is happening because texture\_id does not match any value in grayscale\_plate, and thus material\_key is never assigned.

To fix this, you should ensure that material\_key is assigned a value before it is used. One way to do this is to initialize material\_key with a default value before the loop. Also, it's good to include a check to handle cases where no match is found.”

By ChatGPT-4o

Revised/Fixed code part in file\_utils.py (New lines on 542 and 547)

Lets see if this fixes it, also note, just realised MR might not be shifted\_t yet unlike others.

FIXED!!

ALL 5 Scenes meshes now generated! Only problem now is that almost all have big inverted triangle mesh on top of camera due to occlusion (kitchen have smaller)

Forked [Edgenet360/Mesh-generation from Simon Lisowski](https://github.com/SimonLisowski/Mesh-Generation) to my [GH repo](https://github.com/Muhammad-Hazimi-Yusri/Mesh-Generation) and pushed changes.

Todo tomorrow: rerun everything that don’t have material/depth map in its folder and copy it there for easier reference later. (Output folders in edgenet360/Output/$scenename such as KT,LR,MR,UL and ST)

# Tuesday, 02 July 2024

-prepped for meeting on tablet.

-cleaned the github repo more (also for some reason vscode not liking unity files, so gonna use cursor for that)

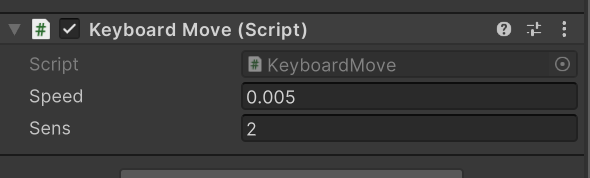
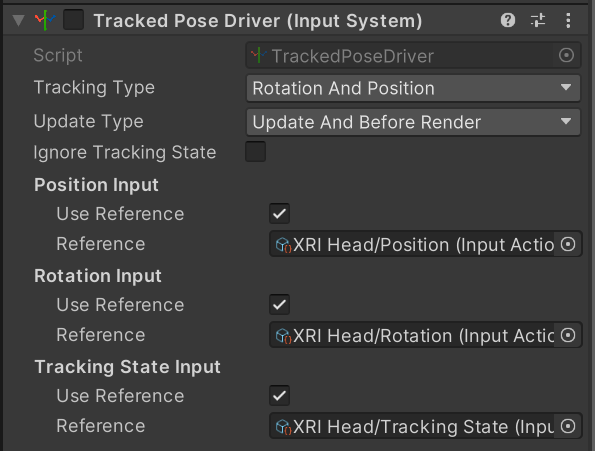
-refactored smoothing index error fix to be more verbose and clear

-generated all 5 scene meshes (.obj and .mtl) and its pipeline intermediary outputs (depth map, material map)

Reminder that the 5 scene meshes are Kitchen (KT), Listening Room (LR), Meeting Room (MR), Studio Hall (ST) and Usability Lab (UL). Note that MR is taken from Living Room in Data/Input folder.

-Tested importing all scene meshes into Unity✅

-Using keyboardmove.cs script made by GDP group to test without VR (there’s also a way to simulate VR with unity addons etc but might as well use this script as its simpler and get the job done)

To use NON VR, on Main Camera, disable Tracked Pose Driver component and enable Keyboard Move script and vice versa to enable VR again.

Tested on my Sony WH-1000XM4 Headphone (wired) and can confirm there is some noticeable difference for the reverb especially comparing Studio Hall to Listening Room etc.

Occlusion is very strong here, as going behind an occluded object sometimes muffles the sound too much imo, maybe because the steam audio setting currently is too cheap (not enough ray bounces?)

There is however a big problem with the occluded part above and below camera being rendered with mesh that sometimes block the sound source origin point thus need to be moved. Better solution is to just remove this anomaly altogether which require further delve deeper into the pipeline codes. Though, maybe this is already in plan but my pipeline running is not correct (unlikely as the same occluded cone area is seen in GDP report Figure 4.23 page 53, on KT scene albeit smaller which is the same output I got as seen previously.

Now im bored so lets test it on VR! 😊

## 2nd Meeting w/ Dr Hansung Kim

-good job me for getting quite a lot done for 1 week, but keep reading! Don’t be lazy to read!

-work on removing the anomaly meshes (top and bottom cone) from occlusion/geometry infinity tingy, the meshes so far looks pretty horrible, especially some of the material assignment.

-will deal with audio later, he gonna send some tools/code. And can ask Mona for help if needed.

Updated TODO:

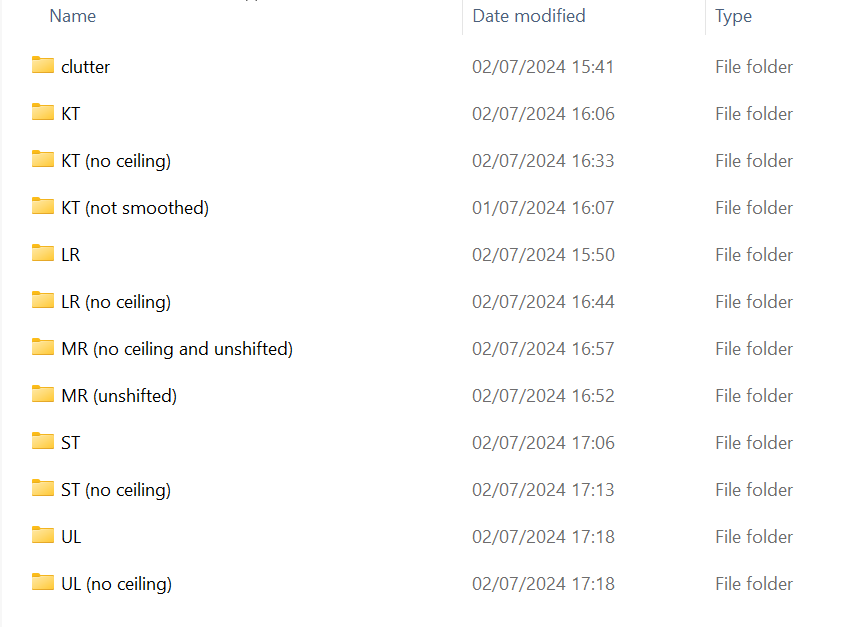
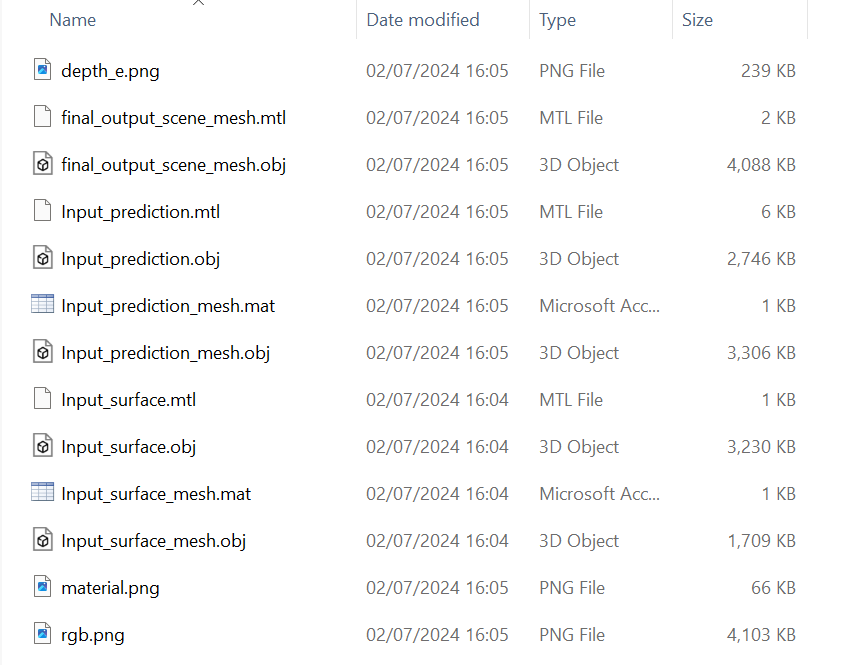
1. Clean up meshes (remove top and bottom cone) done
2. Modify GUI.py to include options to render without ceiling (removing –include\_top y parameter basically on checkbox) done

Note: The voxel obj (not mesh yet) is input\_prediction.obj

Thus, regenerating every scene meshes to include all .obj output in folder.

… ok idk why but the obj im getting for KT now vastly differs from before… ( a lot worse I think)

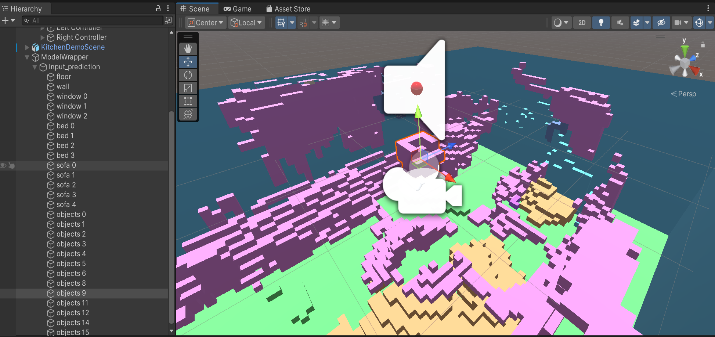
Ok fixed now, maybe I chose the wrong (non shifted image) input.

Alright, generated all mesh for now. Committing and pushing and logging off for today.

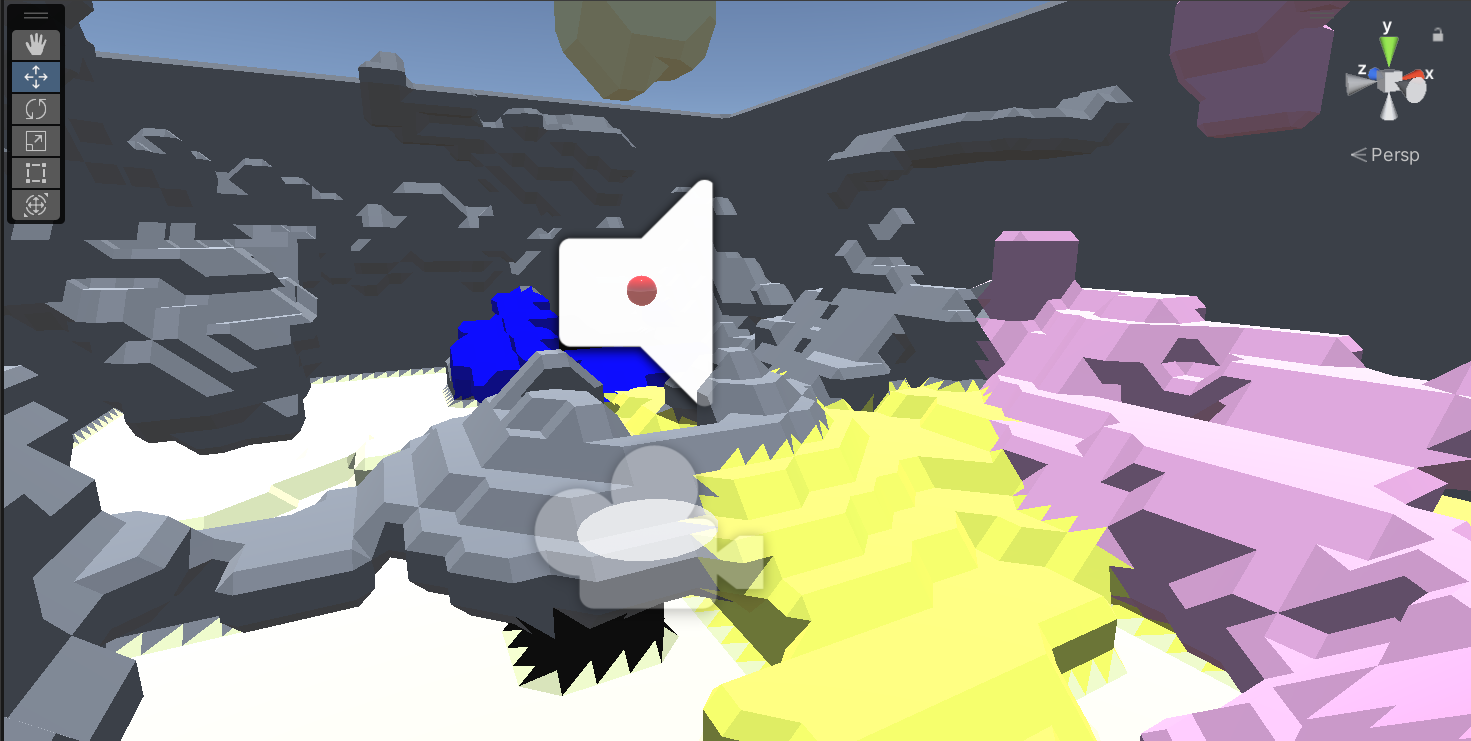
TODO tomorrow: learn about pipeline and where to inject code to clean up top/bottom anomaly meshes.

# Wednesday, 03 July 2024

Time to think how to solve the problem most optimally. Note: Open .obj and .mat on text editor to see how the meshes and material to it are assigned to understand code. Solution I thought so far is either to remove early in pipeline (right after voxel is generated) OR remove after meshes is generated. I think former is easier as some of the meshes/voxel anomaly is linked with non anomalous, especially on floor, and some instances on ceiling.

The voxel representation also have some objects labelled and most of them are unlinked which make it easier to work with, though because of most of the floor anomaly is linked with different objects anyway, maybe easier foolproof way is to just remove certain voxel in a radius around origin (camera point).

Interesting observation, importing all the meshes in blender, especially the final one, shows there’s lots of vertices can be removed for the meshes for optimisation ( I think). Also, because the blender .sobj import don’t have proper material assignment script like import scene implementation in Unity, opening the first intermediary output (input\_prediction.obj, the voxel one) is useless as the material and objects is not linked to the vertices compared to opening them in Unity.

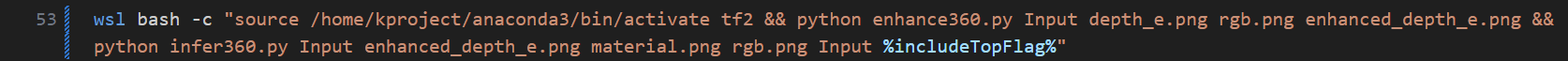


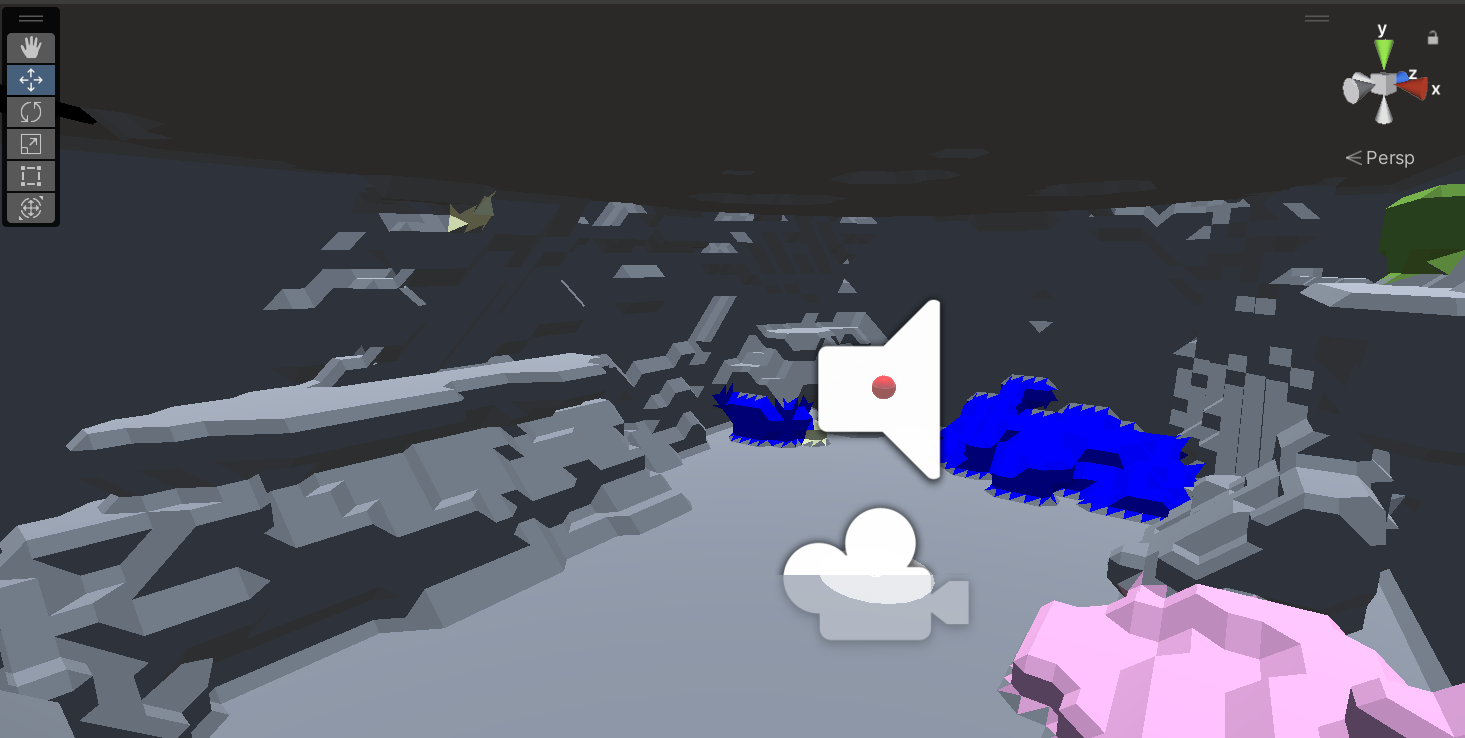
As previously stated, due to incorrect depth prediction, the neighbouring meshes also suffer the gradient depth anomaly where it should have instead be flat, so maybe a better solution is to flatten the meshes and ignore material assignment for the radius near the origin camera point instead of removing them altogether. Only problem with this solution is that the radius need to be big enough to fit the anomaly region but not too big to include other object which can make it lose details.

ANOTHER idea is to post-process depth\_e.png to have ‘infinite’ distance (whiter gradient) on top and bottom so no meshes will be formed. Obviously post-processing using separate script before passing it through the infer360 (edgenet360) would be ideal instead of modifying the monodepth code itself which might be too difficult and introduce more error. Idk why it doesn’t seem to be working, (still don’t know but don’t care, not related) doesn’t seem to have much different, maybe need more aggressive white? Ok idk man, going back to first method of manipulating meshes instead cus this image processing thing is hard.

After discussing with Mona, it seems the enhance360 might be the key, as seen in new\_shifted\_disparity.png, it have white gradient, thus we need to try use is/implement into the pipeline. The problem is that we don’t have shifted\_disparity.png. Thus, maybe we need it from monodepth output? Ok im dumb dumb, we get new\_shifted\_disparity from enhance360.py, to be inferred from. The shifted\_disparity is the depth map which is depth\_e.png we have. So just add another line, which GDP group missed out in their pipeline, which is to first enhance the depth map which tbf, does be confusing, It took me nearly 1 hour to relate that disparity is the depth map lol, especially because 360monodepth seems to have both as different meaning in its code. The example is not enough, should have more example and one without naming and the description of input, sure it have them in argparser arguments, but that’s not working for enhance360.py (it works for infer360.py) for some reason and just throw error lol.  
IT WORKS!

### The solution basically

The GDP group forgot (I think) to enhance the depth map after generating it on 360monodepth before inputting to edgenet360 infer360.py, so I just added another command in the combined.bat to include the enhance360.py process as well. GUI.py intermediary file copy also updated (refer git).

The main problem imo stems from the lack of more clear documentation on the tools, the example provided is confusing and not clear enough. It should have also generic command argument template to CLEARLY show what are the input and output, as the name of the files (shifted.png, new\_shifted\_disparity etc) is confusing and misleading especially when different tools uses different name for similar thing (depth\_map on 360monodepth). The GENERATED MESH (KT w/ ceiling) NOW IS PERFECT IMO, no more cone anomaly and its more structured!!! 

However, for some reason, now no ceiling/top option parameter is not working… It outputted meshes with ceiling regardless of parameter.

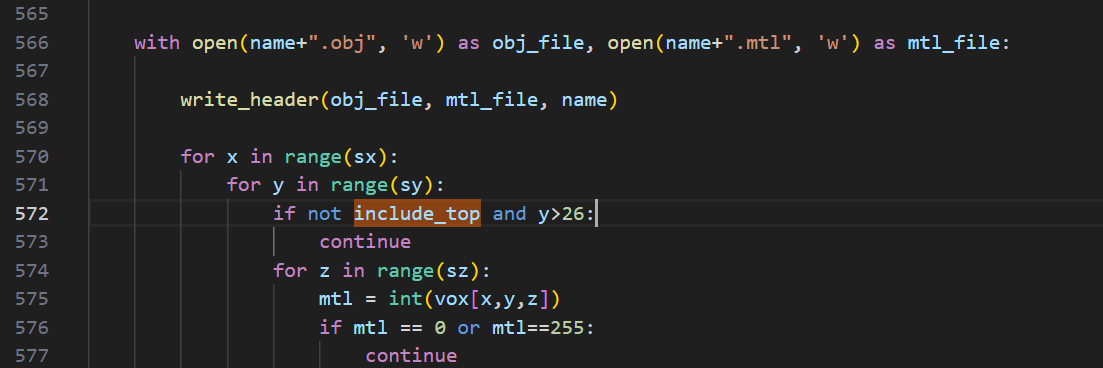
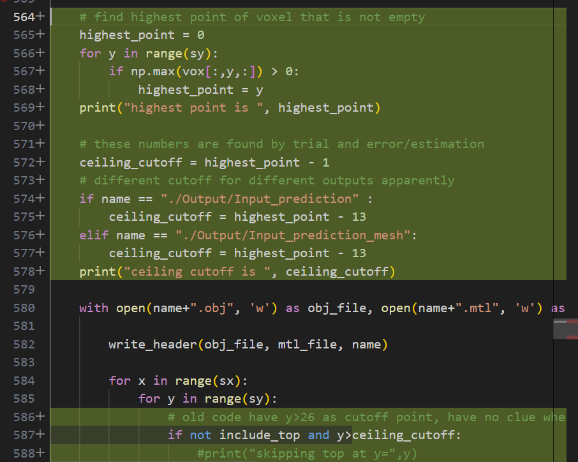
Lesson learnt:

* Even if its tested, but especially if its not fully tested and you haven’t verified it yourself fully as expected, don’t trust it to work flawlessly and there’s no oversight/assumptions from previous group. Especially if your project is different/have different aims.
* If implementing something to fix bug, try check all tools if they have the thing u are looking for and you just haven’t used it (in this case, the enhancement depth map thing), fully understand all tools involved to make sure you are not reinventing the wheel!

TODO next:

1. Figure out why –include-top n not working anymore FIXED
2. Regenerate all meshes output and its no ceiling variant
3. Remove old docker containers automatically to remove docker clutter
4. If provided tools/methods to continue with audio test/eval, do that, if not
5. Maybe make the project/pipeline more easy to reproduce/run by having a .txt for project dir etc so no need to replace all manually on vscode etc

Found the culprit for include\_top y not working in file\_utils.py, however question remains where does y>26 stems from, why 26? changing it to 16 does remove the ceiling but also I think voxel other than ceilings… Is there a better way to remove the linked ceiling object regardless of height, or have dynamic height depending on room height, or maybe get the highest voxel point and -2/3 to get ceiling estimate.

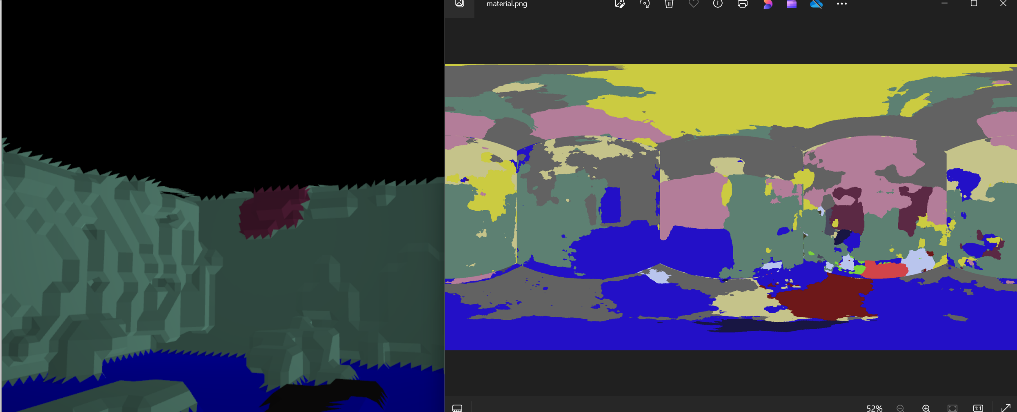
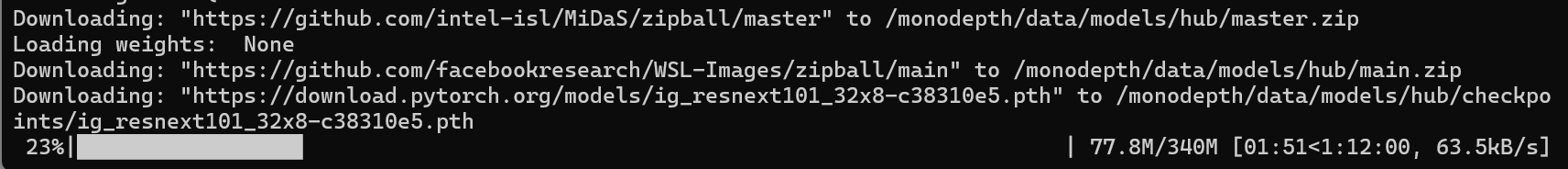


Nvm, easier method is to ignore this, maybe even remove it and use include\_top parameter for custom post\_process.py function to remove all ceiling related voxels etc instead, idk man. OK nvm, for now I just used a dynamic cutoff point thing above (right), seems to be working fine so far (KT and LR w/ & w/out ceiling).

# Thursday, 04 July 2024

TODO today:

1. Regenerate all meshes output and its no ceiling variant
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

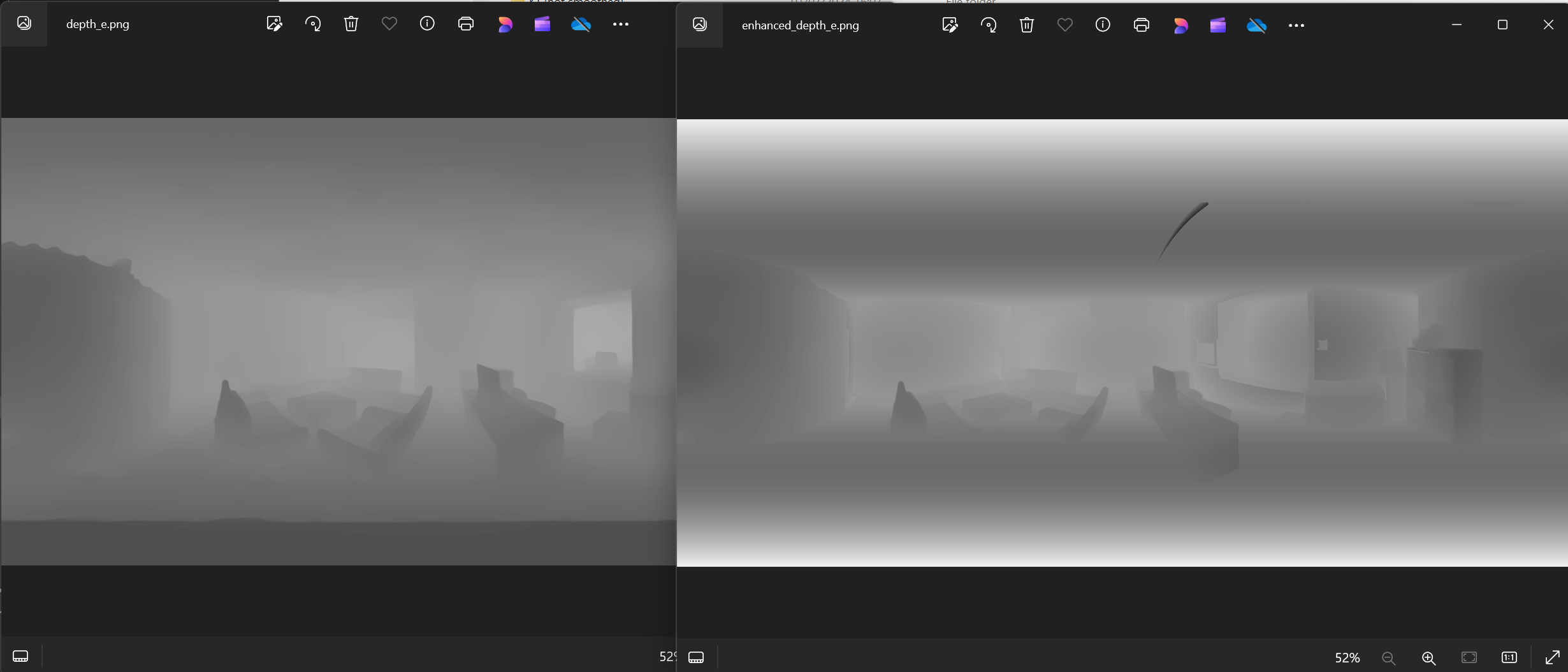
Now time to generate all other scene meshes. But maybe I should find out how to shift the meeting room rgb png first to make better depth map. Note: always remember to run docker desktop if it doesn’t auto start on Windows startup.  
The pipeline being dependent on new instance of docker container every time it runs not only create clutter of containers, but make the process depending on download speed as well…

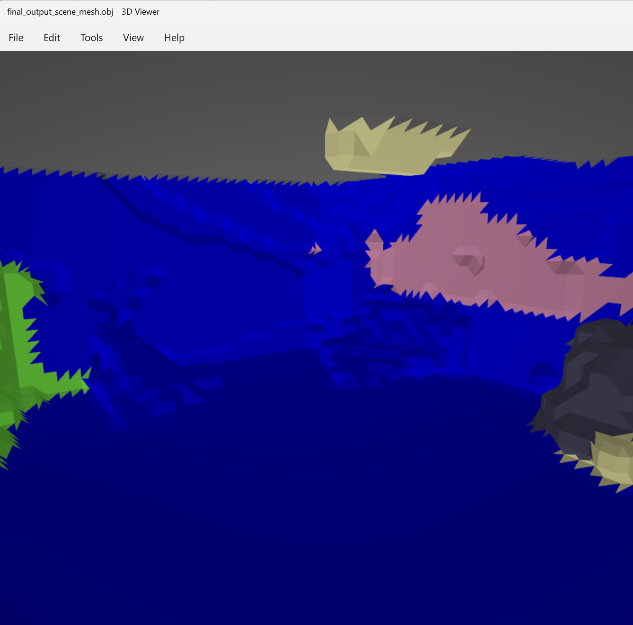
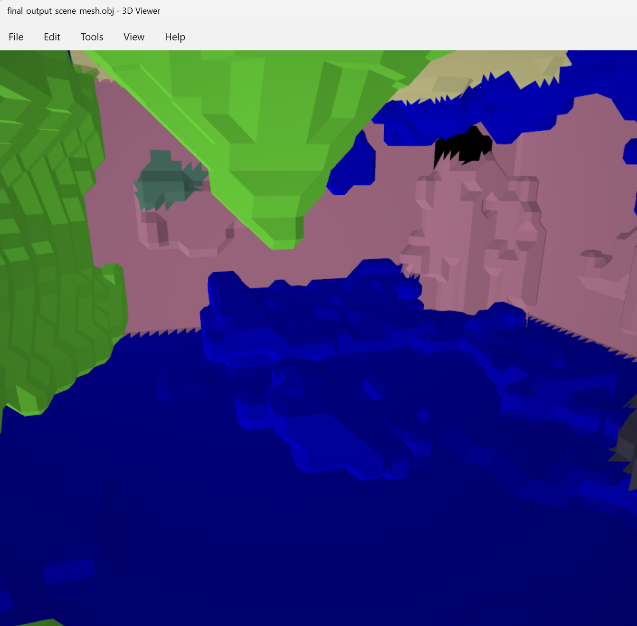
MR ceiling material (black ceiling) for some reason don’t have any material assigned (Default material) even though it was detected as metal (?) Is it because its not shifted first? Yes, shifting fixes it!

For some reason, every time I ran it, the output is different, sometimes vastly different imo.

OK, ST works fine so far (correct ceiling material and very accurate imo) so most likely the reason is just because its not shifted. Then maybe include shifting into pipeline and have checkbox for images not shifted yet so it can be automated as well. ST w/out ceiling also works. Now I think about it, the only reason no ceiling is useful if u don’t import to unity via import scene, cus if u do, u can just disable ceiling related mat items, not perfect but work fine enough to see top down.

OK after reading the Edgenet360 paper, kinda makes sense why the GDP group don’t use enhance360 as it was built for stereo depth matching preprocess in mind. However, there also other important step like gradient to white on top and bottom edge which also needed for the normal depth map output to prevent anomaly. However, using it comes with some drawback of sometimes losing details (some important and other time not). For example, in this UL depth map:



The left sofas got cut off too early, however, the enhanced also recognise the mirror reflection and ignored it perfectly, so it’s a compromise, I think maybe having the depth map from this monodepth to be more black to have more ‘depth’ could fix this, or having different threshold (is this the –baseline option in enhance360.py ?) as seen in this comparison of enhanced vs not output .obj:

As seen, enhanced (left) is more clean but considerably emptier (esp near centre) as the depth map enhancement is built with pre processing much noiser and ‘stronger’ stereo depth matching in mind, not cleaner/smoother and ‘weaker’ monodepth output.

UL w/out ceiling is not cut short enough, changing -13 to -16 to be more conservative for more height and variables (wall/ceiling details be damned), ok maybe-16 too much, lets increment little by little. Go back to -14.

Updated TODO:

1. Add shifting to pipeline option done and fixes material assignment bug.
2. Regenerate all meshes output and its no ceiling variant done (at least for enhanced depth map, not including TODO 3 modification)
3. Make monodepth depth map ‘stronger’ to work with enhance360 better.
4. Remove old docker containers automatically to remove docker clutter
5. If provided tools/methods to continue with audio test/eval, do that, if not,
6. 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

Shifted MR doesn’t seem much different tbh for the meshes as expected, showing that ‘stronger’ depth map mod is needed still, also change from -14 to -16 ceiling cutoff substraction because MR is weird like that…

But shifting FIXES the default material bug as expected because shifting makes the alignment of cube map projection in which material detection/assignment (DBAT) better.

For TODO no. 3, maybe I can modify enhance360 parameters or add some code lines/change some part to fit monodepth better? (require in depth understanding of enhance360 and related codes)

OR change parameter of monodepth so that it output ‘stronger’ depth map?

Unrelated future work IDEA: add more sounds option/better UX/UI for it on VR/desktop interface so can hear difference better instead of only 3 different sound source rn (speech, gunshot, music)

I want to try work on TODO no. 3 but my brain is fried rn, so enough for today!

# Friday, 05 July 2024

WFH, not much recorded, mostly reading papers and testing to pull the repo to see if it works. TLDR: It doesn’t yet, need to figure out submodules and also git LFS for the LiDaR file as that is needed for Unity script to work, or just modify the script to exclude that. But ML pipeline is so broken as the submodules are not pulled together. Another thing is that the repo is too big (around 8GB), need to remove bloat.