How To Run

This has been developed in a ubuntu 20.04 version, although it should be compatible in other OS. For this, you will need: freetype, cmake, eigen3 and tinyply.

The *run.sh* bash script is a simple shortcut for easy execution.

In a new environment you should run first:

- *run.sh install* this is experimental but it should install the tinyply.
- *run.sh init* an alias for *cmake ../* and build

This will open a basic scene of the program.

Next you can't use the script to compile *build*, clean *clean*, and run *run*.

The script also includes some flags to call the other functionalities.

- *run.sh run MODEL −−LOD N* Calculate the N LOD from the input model.
- *run.sh build TILE_MAP −−cm OUT_FILE* Build and calculate the visibility map and save it in the output file.
- *run.sh run MODEL −c 0* Run a empty scene with the model selected and don't use any culling policy.
- *run.sh run TILE_MAP −c 6 VISIBILITY_MAP* Run an scene with the tile map selected and use the visibility map.
- *run.sh* $create_{a}ll_{M}useum_{L}ODs$ Create all the museum LODs.

The patter it's always the same *run.sh* followed by a command and the flags input models in any order, only respecting the flag dependencies (e.g. culling policy always goes followed by it's number). Also the script is just a lot of switch case so it can be read easily.

The script it's just for simplicity but you can also use CMAKE and run it with the flags noted in the *main.cpp* file.

Done List

- Session 1: All done. Although they are arranged in a 2xN line.
- Session 2: Al done. There are many maps but you can try *big_musuem* and *small_musuem*.
- Session 3: All done. The advanced it's done with a octree but for simplicity it only creates one simplification.
- Session 4: Basic done. I don't do the normal clustering.
- Session 5: All done. The hysteresis have two components, temporal and spatial. You can decide the elapse time between changes and the proportion of messes to change.
- Session 6: Basic done. Following the tile map it computes the visibility from the objects in all views.

Note

This project has been developed along the FRR lab and includes some visibility algorithms.