ULR Asset Streamer and CPU Upload

main   **streamChunk**   **streamCamera**   **inputCamera**   **rgbdComponents**

new → **assetStreamer**

loadConfig(stream/config.txt)

new → **assetStreamerThread**

loop [try to load chunks/cams]

> In the mean time, main thread will check if cameras where added
> by background threads.
> In AssetStreamer::update, cameras will be move from addedCameras list
> to cameras list four by four (to release the lock)

> go through all chunks and find one that is not loaded yet
> go through all cams and find one that is not loaded yet
> * it will look for cameras to upload to CPU only in the first maxImagesCPU cams
> * list of cameras is sorted in AssetStreamer::update
> * there is no cpu unload (~10GB catal)

opt [if chunk to load]

loadFromDirectory(chunkFolder)

> load chunk mesh
> load chunk.dat (chunk offset, rot, camList)

loop [for each cam in chunk]

loadFromDirectory(incam.bin)

loadFromBinary(incam.bin)

> incam.bin has the cam's calibration

> at this point all cams in chunk have their calibration loaded

loadVoxels(streamFolder, chunkVoxelsBin)

> load voxels.bin (grid data like scale, origin pos, etc)
> and populate map of voxelCoord -> list of cams
> * map is empty if there were no
> local meshes when running mkchunk

addCameras(camsInChunk, chunkTransform)

> add streamCameras to the addedCameras list

opt [if camera to load (taken from cameras list)]

onStreamCPULoading()

> it will load both color and depth img (in CPU)