Unity ULR Texture Upload

main | **textureLoaderDiscriminator** | **ibrEvents** | **textureLoader** | **viewChangedEmitter**

main → textureLoaderDiscriminator: OnEnable

textureLoaderDiscriminator → ibrEvents: SubscribeToOnViewChanged

main → textureLoader: OnEnable

textureLoader → ibrEvents: SubscribeToOnProposedCamerasChanged

loop [main lopp]

main → viewChangedEmitter: update

viewChangedEmitter → viewChangedEmitter: CheckCameraUpdate

viewChangedEmitter → ibrEvents: ViewChanged

ibrEvents → textureLoaderDiscriminator: UpdateCameras

textureLoaderDiscriminator → textureLoaderDiscriminator: SelectCamerasBySIBR

> **Note:**
> there are some other methods to use
> currently using "SIBR":
> distanceVal = d(i, n) / ref [0,1]
> angleVal = 1 - (1+dot(i, n))/2 [0,1]
> angleVal = (angleVal <0.5) ? ing : angleVal
> cost = (1-a)distanceVal + a*angleVal; a=0.1

textureLoaderDiscriminator → ibrEvents: ProposedCamerasChanged

ibrEvents → textureLoader: UploadTextures

textureLoader → textureLoader: LoadTexturesFromDisk(dataset, selectedCams)

> **Note:**
> It will sort the list of cams using the cost
> that was previously calculated
> It will also calculate what cams are new
> and what are not required anymore

textureLoader → textureLoader: SwitchTexturesCoroutine(toUnload, toUpload)

> **Note:**
> It will only unload cams if it's running out of space
> Cams upload is done in a coroutine to not block main thread

textureLoader → ibrEvents: LoadedTexturesChanged

> **Note:**
> Previous voxel system was subscribed to this event