System test case - #4 _2

Test case details

Tester: Veikko Svanström

Date: 7.4.2025

Device: Laptop computer, low-end, Windows 11 Home V.24H2 x64

Environment:

- Most recent main branch
- Most recent commit is f3e03fe9619d77b65d5a694d6927f71fff807866
- Unity editor version 6000.0.35f1

Test details

This system test is exploratory in nature. The simulation is run via unity editor with the purpose of testing the performance of the simulation with the new features impacting it, mainly the simulated ar / pointcloud mesh. Additionally, the unity profiler will be used to identify performance bottlenecks in the code. The profiler itself slightly decreases performance, which should be taken into account. The tester will see if performance problems still persist when the mesh is visible, and if it is solved via toggling the visibility of the mesh off.

Test steps and results

1. Open the Unity Profiler window from Window -> Analysis -> Profiler and turn on Deep profiling by pressing “Deep Profile” in the profiler window
2. Start recording with the profiler with F9 or by pressing the record dot so that it turns red.

3. Start the simulation by pressing play on top of the unity editor
4. Run and look around the simulation in order to simulate the ar mesh being created

a. Result: Points were scanned and simulated mesh created. There are significant stuttering/frame drops on the testers low end machine when running and looking at the ground (where simulated mesh is being created), regardless of whether the mesh is visible or not.

5. End the simulation by pressing the end button
6. Open the profiling window and view the profiling results.
    a. Result: The profiler shows heavy spikes on CPU usage, by viewing the spikes on the timeline the biggest bottleneck seems to be SeenMeshManager.AddTrianglesToSeenList. 1 execution of this function takes approx. 15ms. Digging deeper into the hierarchy tab we see that Mesh.get_tirangles and mesh_vertices are the largest bottlenecks of this function. These calls take a long time and allocate a lot of memory.

The tester decided no further testing is needed at this point, since it seems clear that there is a performance issue related to mesh creation and updating. The constant lookups of mesh triangles and vertices are costly, because at least for the terrain mesh these lists are huge and they are fetched needlessly multiple times. Issue #107 is created to solve performance issues by using caching for the mesh triangles and vertices. This test case should be run again with the profiler and without it when a fix is implemented.

Conclusions

Issue #107 created.