Stress Test on Scene Graph of DS Engine

Victor Shu

This report is about the stress test on the scene graph of DS Engine. I want to test the upper limit of the number of entities the DS Engine can have and see the performance through the framerate.

I tested against the number of entities from 1 to 10000 and recorded the framerate. These entities share the same mesh and material to prevent using too much memory space. In the test game, there is only a camera with a skybox and entities with default material and a script that makes the entities rotate to simulate a real game environment.

I used control variables in this test. The variables I controlled are planar/hierarchical, opaque/transparent, and VS2017/VS2019. In the test, there is a control group which uses planar, opaque and VS2017, and other three groups that only changing one variable.

The data I collected is listed in the following table.

	VS2017			VS2019
	opaque		transparent	opaque
	planar	hierarchical	planar	planar
1	3105	3080	3030	3120
2	3100	3050	3030	3100
5	3070	3040	3010	3070
10	3060	3020	2960	3050
20	3035	3010	2955	3045
50	2960	2950	2830	3000
100	2635	2450	2220	2700
200	1535	1100	1120	1600
500	590	245	405	620
1000	245	65	180	260
2000	96	16	75	97
5000	25	3	20	25
10000	8	1	6	7

The meaning of the variables:

1. Planar/hierarchical indicates that whether one entity is parented with another one. For example, in a planar setting, object A is a child of nothing, object B is a child of nothing and object C is a child of nothing. In a hierarchical setting, object A is a child of nothing, object B is a child of object A and object C is a child of object B. This can test the impact on performance caused by the world matrix calculation.
2. Opaque/transparent indicates that whether the entities are transparent. The engine sorts the transparent objects to prevent one transparent entity seeing through another one. This can test the impact on performance caused by sorting. The transparent objects are just declared to be

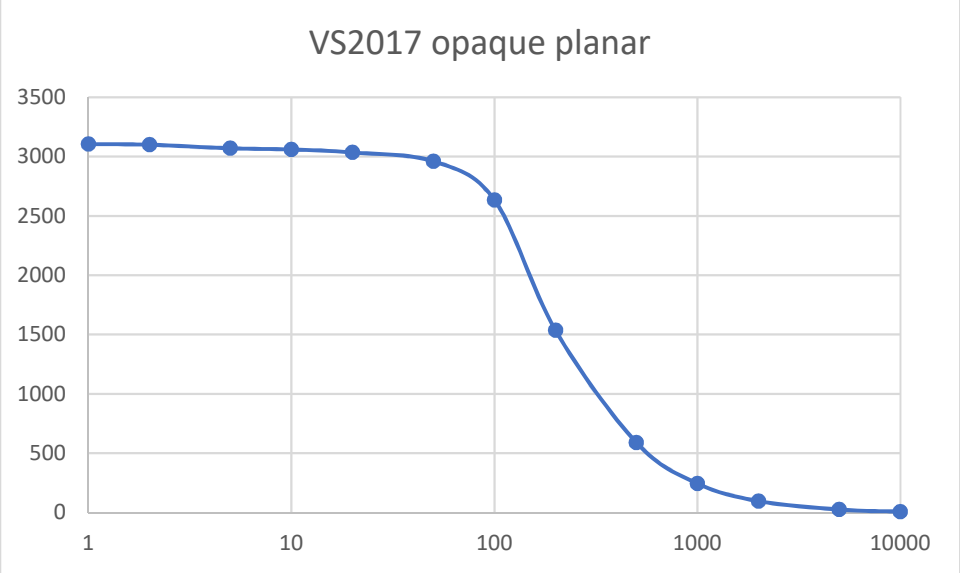
transparent, but they are neither actually transparent nor the alpha blending state changed so no other factors are introduced to the rendering part. The sorting algorithm is from C++ STL.

3. VS2017/VS2019 indicates that whether the engine is compiled with Visual Studio 2017 (MSVC CL 19.1) or Visual Studio 2019 (MSVC CL 19.2). This can test the impact on performance of the compiler. Only the compiler of the engine itself is changed, no 3rd party library is affected.

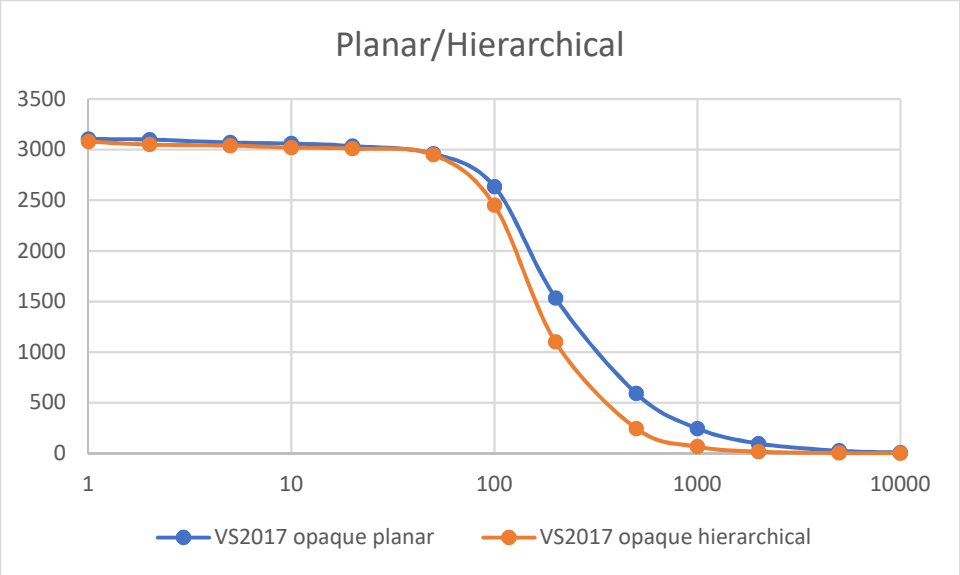
The test machine/environment/program specification:

- CPU: Intel Core i7-8086K 4.0GHz
- GPU: NVIDIA Geforce RTX 2080
- Memory: 2x16GB DDR4 3000MHz
- OS: Windows 10 Pro 18890
- Compiler: Depends
- Configuration: Release (Maximum Optimization)
- Resolution: 1366x768

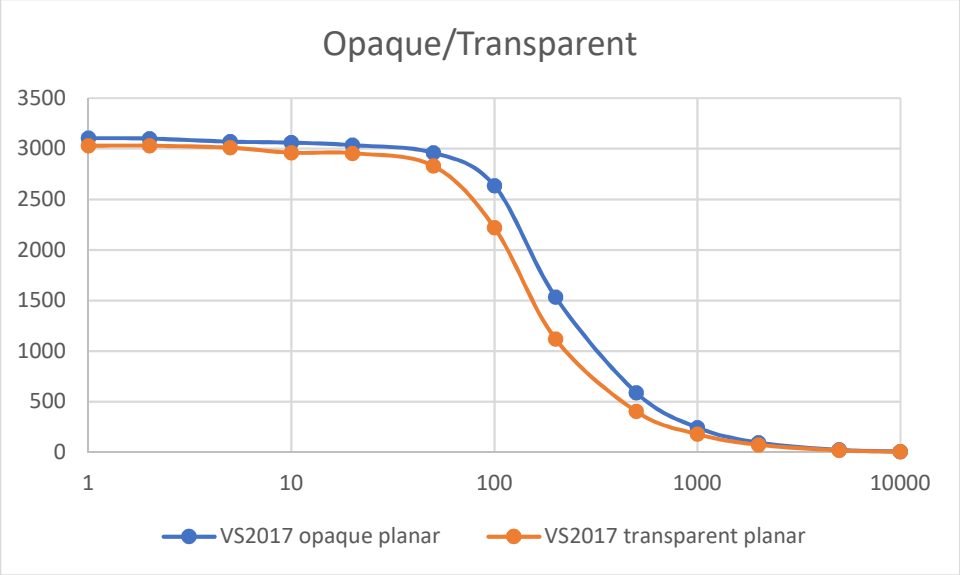
Analyzation:



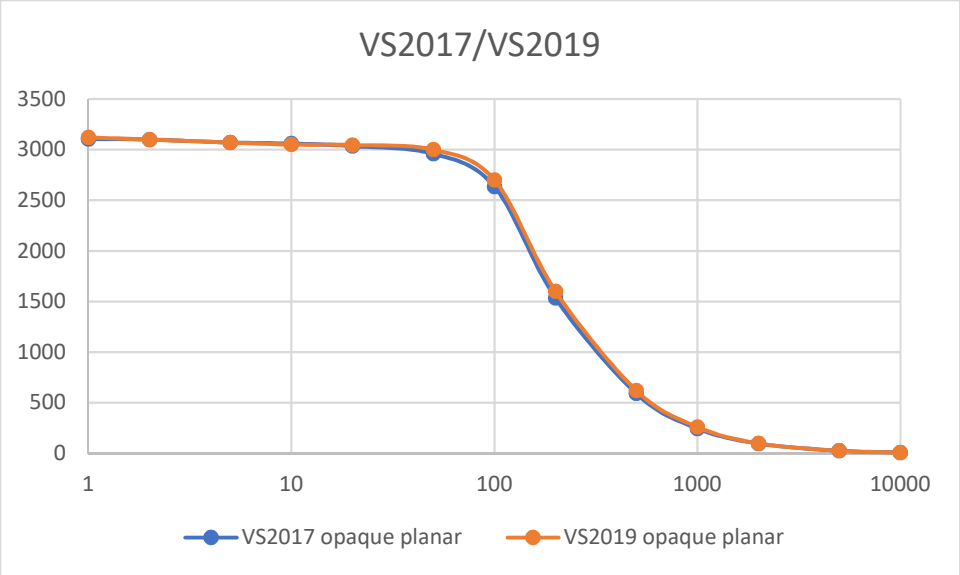
It's obvious that with the number of entities growing, the framerate drops. When the number of entities is around 50 to 100 this process becomes more dramatic. When the number goes up to 2000, the framerate dropped below 120 even on a high-end machine which can be problematic.



The world matrix calculation does take an amount of time. From the chart it's obvious that the hierarchical setting is always slower than the planar setting. The impact started to get obvious at around 200 to 500 entities. However, the impact is not so dramatic, so this proves that the DS Engine is good enough at doing hierarchical world matrix update.

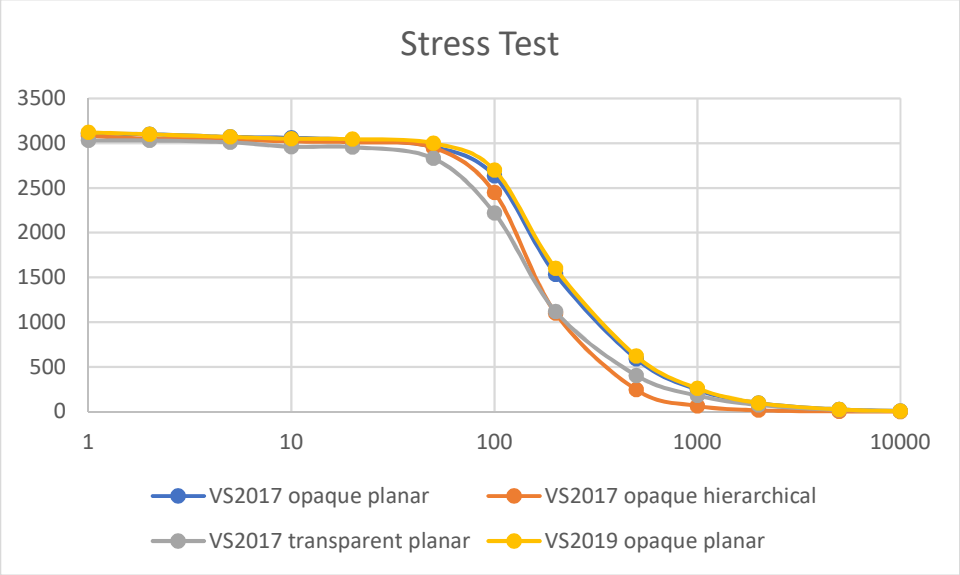


Sorting also takes time. However in the test the objects are added in an reversed order which means it's the worst case for the sorting algorithm. The performance impact is fair compared to the hierarchical setting. And there are not always thousands of transparent objects in a game.



The Visual Studio 2019 compiler is faster than 2017. There's not much a compiler can do for a single thread program, but this still shows the capability of the new compiler. With all 3rd party libraries compiled with the new compiler the performance gain can be bigger.

Conclusion:



This test shows the capability of the DS Engine Scene Graph. To be honest, I'm happy with these results. It shows that even it's single-thread, the DS Engine Scene Graph can still process things fast enough. If we can add multicore and partitioning to the engine, I believe that it can be even faster.