

Stress Test on Physics Engine

Scratch Engine

Test Environment

CPU: Intel(R) Core(TM) i7-6700K @ 4.00GHz

RAM: 24.0 GB

Brief Introduction


The test focuses on the physics engine, especially the dynamic bounding volume hierarchy, the core data structure within the engine that is capable of separating and grouping objects based on their position. The data structure is based on a full binary tree, therefore giving $O(N \times \log M)$ performance in the best scenario where N is the number of moving objects and M is the total number of objects.

Test Setup


Usually, there are not really a lot of moving objects in the engine at a given frame. Therefore, I set N to be 100 in all tests. All other systems are turned off during the test process in order not to introduce additional biases. However, the pipeline remains the same where there are still 3 threads running and one of them is running only with physics engine. The performance is measured through the fps counter on the head bar which is updated each frame. The scene is set up randomly. All objects are randomly distributed in a $100,000 \times 100,000 \times 100,000$ space and they all have a random size of 1 to 10. All objects are using box colliders which is the most expensive one in the system. Also, all tests are conducted under release mode and involve no GPU calculations.

Test Result


$N = 100, M = 100$

 DX11 Game Width: 1600 Height: 900 FPS: 10417 Frame Time: 0.0959969ms DX 11.0


$N = 100, M = 1,000$

 DX11 Game Width: 1600 Height: 900 FPS: 1152 Frame Time: 0.868056ms DX 11.0


$N = 100, M = 5,000$

 DX11 Game Width: 1600 Height: 900 FPS: 172 Frame Time: 5.81395ms DX 11.0

$N = 100, M = 10,000$

 DX11 Game Width: 1600 Height: 900 FPS: 99 Frame Time: 10.101ms DX 11.0

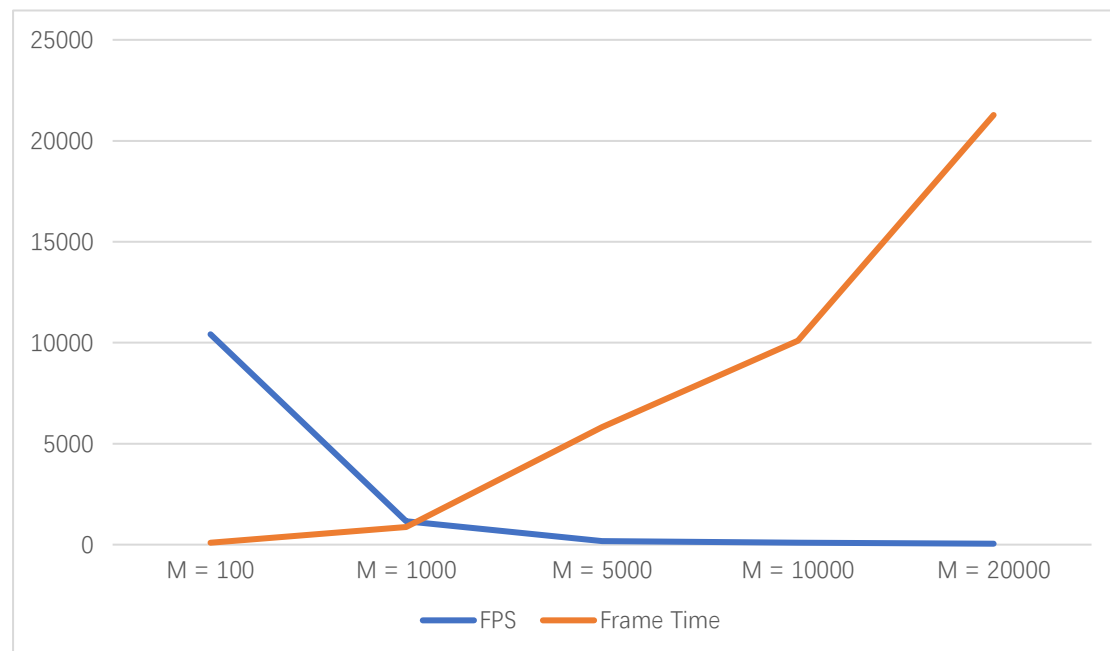
$N = 100, M = 20,000$

 DX11 Game Width: 1600 Height: 900 FPS: 47 Frame Time: 21.2766ms DX 11.0


Discussion

The result is pretty delighting. The physics engine runs at around 100 fps with 10,000


objects. The results seem to indicate a logarithmic relationship between M and the overall



performance. In fact, the actual performance is largely determined by the actual number of objects collided. For this, I made a clustering test where I put everything colliding with each

 **DX11 Game** Width: 1600 Height: 900 FPS: 38 Frame Time: 26.3158ms DX 11.0

other. In this test, I only set M to be 1,000 and the fps count was decreased drastically to 40 since everything was colliding and the calculation between orientated boxes took a lot of efforts. For comparison, I made another test where I added extra objects which were far

 **DX11 Game** Width: 1600 Height: 900 FPS: 85 Frame Time: 11.7647ms DX 11.0

away from the moving objects. For this test, I put 20,000 objects and 10,000 of them were far away. The performance did not drop too much compared to the previous test where M was equal to 10000, which indicates the data structure does well on separating objects that are far away from each other.

Conclusion

As shown, the system works as expected on separating objects so that the engine can cull out a large number of objects and only spends time working on objects that may collide with others. However, the system will not help much when there are a huge amount of object clustering together. Also, there should be spaces of improvements that can be done in the future to further improve the performance.