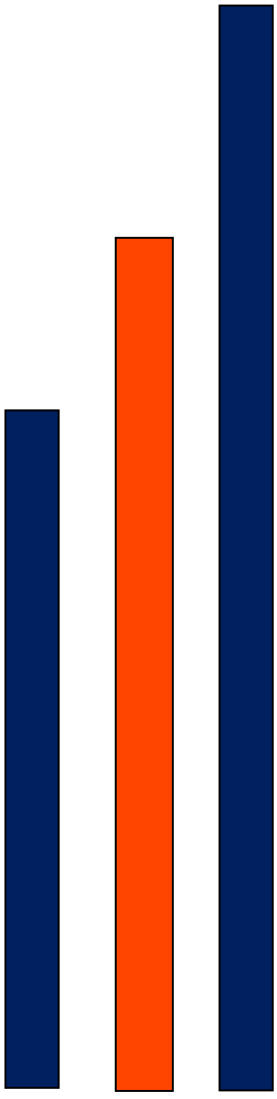
Vulkan Rounded Cell Particle Detection.



User Guide

Jackie Bell

3, 2024

Table of Contents

1 Report of 2024 05 13

The compute shader which was taking 24.15 ms for 1M particles now performs the same in 5.09ms. The improvement results from fixing a recursive bug. (Compare fig. 1,fig. 2)

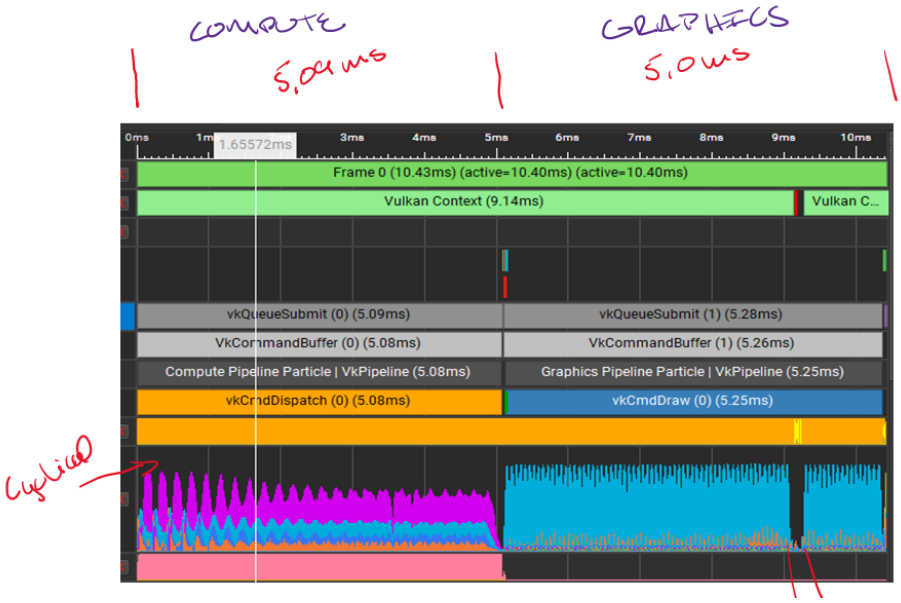


Figure 1: NVidia Insight GPU trace showin the proportion of compute to graphics.

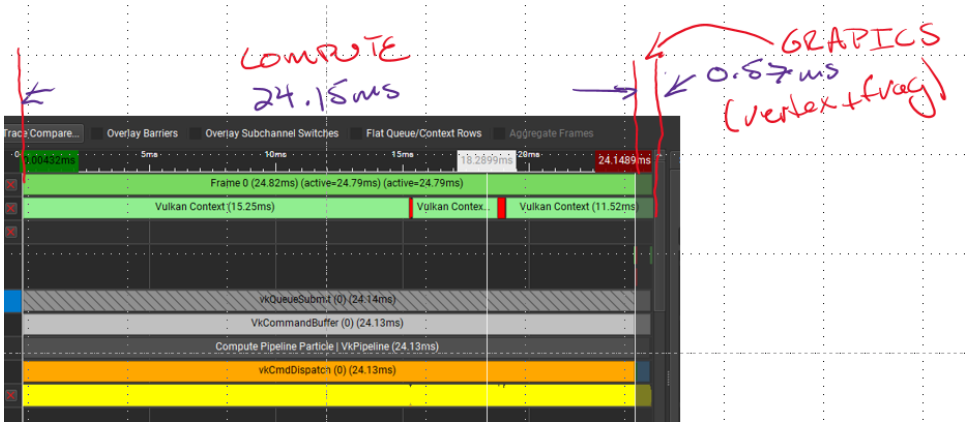


Figure 2: May 2nd NVidia Insight GPU trace showin the proportion of compute to graphics.

Fig. 1 also suggests possible further performance enhancements. The compute is cyclical and not fully utilizing the GPU. There is a large break in the graphics pipeline. Both compute and graphics pipelines are not yet fully threaded.

Table 1 shows improvement across the board (with reference to table 2). One million particles are now processed at 78 fps. Fig. 3 is greatly improved and appears linear. The red

line is a linear fit. The green line is $0.5E-8x$ where x is the number of particles. I am trying to display the linear line. The fit line is above this so the relationship is super linear?.

There are two types of density. The first is the particle density which is the ratio of the number of particles in a cell to the maximum number of particles that can fit in a cell $\frac{cell\quad population}{max\quad cell\quad population}$. The maximum number of particles that can fit in a cell depends on the radius of th particles. In the current case the *max cell population* is 30. The collision density is the number of collisions divided by the number of particles in a cell $\frac{number\quad collisions\quad in\quad cell}{cell\quad population}$.

I have included a section on how the bench-marking/verification is done. One of the problems I had in reviewing particle collisions detection is that the number of collisions in the data was not verified against the number of collisions the method detected. I do this verification so the numbers here are starting to have credibility. although I am not sure I understand them completely.

Table 1: Performance by number of particles and collisions where max. cell population is 30.

Particles	Collisions	fps	Col. Density	CPP	Linearity
1025	479	2016	0.5000	2.1399	4.839e-07
17409	8127	1673	0.5000	2.1421	3.433e-08
33793	15775	1544	0.5000	2.1422	1.917e-08
50177	23422	800	0.5000	2.1423	2.491e-08
82945	38710	551	0.5000	2.1427	2.188e-08
115713	53999	444	0.5000	2.1429	1.946e-08
246785	115167	396	0.5000	2.1428	1.023e-08
508929	237503	116	0.5000	2.1428	1.694e-08
754689	352191	76	0.5000	2.1428	1.743e-08
1000449	466879	115	0.5000	2.1428	8.692e-09
1246209	581567	51	0.5000	2.1428	1.573e-08
1508353	703903	78	0.5000	2.1428	8.500e-09

Table 2: May 2nd Performance by number of particles and collisions

Particles	Collisions	fps	Density	CPP	Linearity
31	14	1716	0.5000	2.2143	1.880e-05
257	125	1994	0.5000	2.0560	1.951e-06
4097	1918	482	0.5000	2.1361	5.064e-07
32769	15296	158	0.5000	2.1423	1.931e-07
117650	54908	63	0.5000	2.1427	1.349e-07
287497	134168	27	0.5000	2.1428	1.288e-07
551369	257306	16	0.5000	2.1429	1.134e-07
970300	452810	10	0.5000	2.1428	1.031e-07
2299969	1073324	4	0.5000	2.1428	1.087e-07

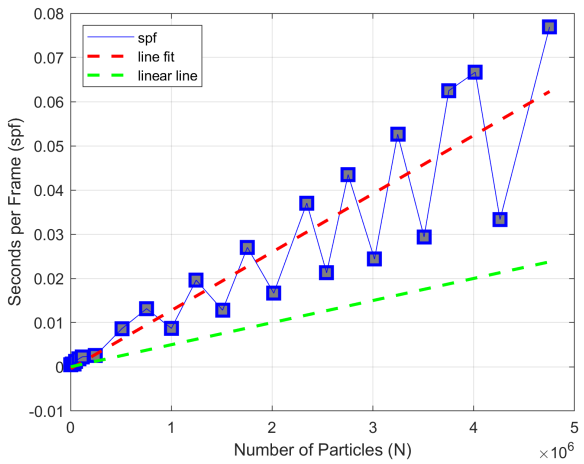


Figure 3: Seconds per frame for 0.5 collision density with 30 particles per cell verses number of particles.

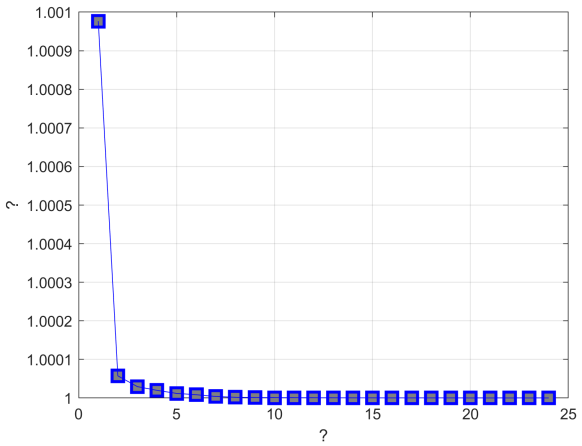


Figure 4: CCP number of collisions divided by number of particles for 0.5 collision density with 30 particles per cell verses number of particles.

2 Benchmarking

2.1 Matlab testing and verification code

The verification procedure is a three part process. Additionally, the testing is performed over two sets - one for a varying number of particles at the same collision and particle density, and then at a constant number of particles and particle density but where collision density varies from 10 to 100%. For each benchmark file the *rccdApp* will test for performance and reliability.

2.2 Generate Benchmark-Verification Data

A Matlab application was developed to produce benchmark data and is abbreviated *rccdGen*. The data is *light weight* as it only requires downloading code as opposed to transmitting very large datasets. The *rccdGen* application provides facilities for selecting different configurations of data. A *benchmark-set* data file is a list containing the compute kernel work-groups, `vkSubmit(..)` dimensions, number of particles, particle and collision density, for a number of data files. The benchmark-set file contains a large number of configuration lines so the application allows selection of only those lines required for testing.

2.3 Run Benchmark data

The *rccdApp* is run in verification mode where it iterates over each benchmark file in a loop. *rccdApp* reads the data from each configuration file, loads the benchmark data, creates a results file, and runs the test. Each test runs for 60 seconds recording the number of particles processed, the number of collisions detected, and the frame rate for each second. If the number of particles or collisions do not match the contents of the benchmark file an error field is written to the results file.

2.4 Analyze the data

Another Matlab application, abbreviated *rccdVerf*, is run over the results data collecting parameters and checking for errors. This data is coalesce into frame rate v number of particles, time per frame versus number of particles at the same collision density and frame rate versus constant number of particles over varying density, Fig. **??**, Fig. 3, fig. **??**, and result Table 1, **??**