OpenCL / OpenGL Particle System

Jack Schofield
06/07/2022

Background

This project aims to analyze the performance of a particle system that utilizes OpenCL. A 3D simulation is written in OpenGL that spawns a group of particles over a series of objects. Physics calculations are performed to have the particles fall around these series of objects (and each other), which is where OpenCL is used. The particles fade to black and disappear, using shader code (written in GLSL).

Our independent variable is *NUMPARTICLES*– which is simply the number of particles generated in the 3D simulation. Dependent variables are GigaParticles/sec (how many billions of particle calculations are being performed each second), and the frame rate the simulation runs at.

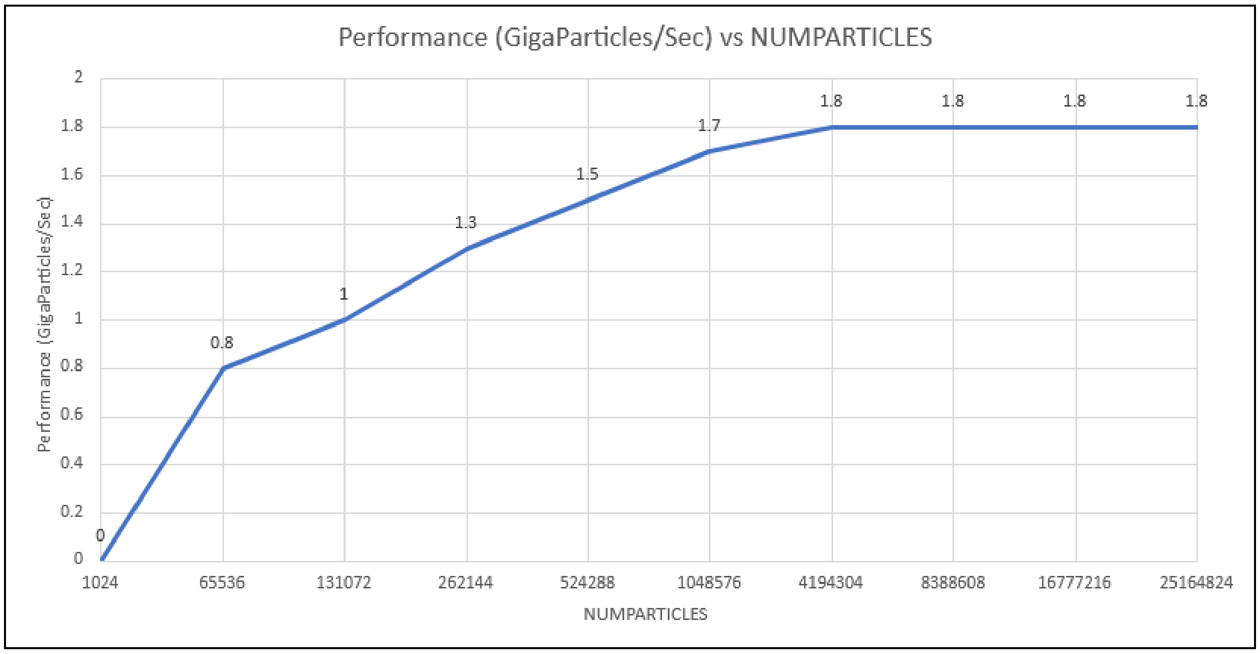
This project was run on my personal computer, which has an AMD Radeon RX 570 (4GB VRAM) and the most recent build of Windows 10.

Performance Results

Table

NUMPARTICLES	Performance (GigaParticles/Sec)	Frames/Sec (Low)
1024	0.0	1380
65536	.8	1230
131072	1.0	1020
262144	1.3	930
524288	1.5	680
1048576	1.7	460
4194304	1.8	143
8388608	1.8	79
16777216	1.8	41
25164824	1.8	27

Graph



Commentary

From the performance graph, I'm noticing what seems to be a hard limit of 1.8 GigaParticles/Sec on my machine. I pushed the program to the limit, with the largest *NUMPARTICLES* values I could before crashing, and could not get over 1.8. In some form, my GPU is being pushed to its limit, either in its bandwidth for sending/receiving data, or its ability to compute the massive amount of data being sent to it.

I'll also note that performance increases non-linearly with *NUMPARTICLES*. For example, the increase from 524288 → 1048576 only saw a 1.133x increase in performance.

What this means for parallel computing is that once we are getting close to the limit, the returns begin to diminish. This is especially noticeable for values over 4,000,000. At this point, we can see the framerate pretty much get cut in half every time we double the *NUMPARTICLES* value, but the performance is staying the same.

Now, for game development, this means that you need to balance your fidelity with real-world performance. Frame rate will scale (close to) linearly with *NUMPARTICLES* at high amounts, but as mentioned earlier, the same cannot be said for the compute ability of your graphics card– the returns will eventually become diminishing.

As evidence for this, let's study this snippet of the table above:

NUMPARTICLES	Performance (GigaParticles/Sec)	FPS Low
1048576	1.7	460
4194304	1.8	143

The jump from 1048576 to 4194304 sees 1.05x the amount of particles computed (per second) for 3.2x the performance hit. At this point, the returns are extremely diminishing, and the program is getting much more resource-hungry. While 143 FPS is not bad by any means, when placed into an entire game environment this single visual effect would quickly become a noticeable performance hit due to its poor optimization.

To conclude, in short, parallel computing is all about finding a balance.