

Benchmarking Real-Time Particle Rendering on Commodity GPUs

Xiao Yao

[affiliation]

Layout demonstration — sample record, not a published paper.

Abstract

A reproducible benchmark design for real-time particle rendering: a workload defined by particle count, a frame budget written as update plus render cost, and a protocol that reports median and tail frame times with the files needed to repeat the run.

Keywords: real-time rendering; GPU benchmarking; particle systems; reproducibility

1 Introduction

Real-time particle systems sit between simulation and rendering: cost grows with particle count, but the frame budget does not. A benchmark is useful only when another laboratory can repeat it, so this specimen foregrounds the workload definition and the reporting rules before any number appears.

The target is a commodity desktop GPU rather than a research cluster. That choice makes the numbers less impressive and the protocol more interesting: thermal state, driver version and background load all have to be written down.

The record is written for a reader who wants to run it. Every section answers a question that reader would ask in order: what is measured, how the workload is fixed, what is reported, and what would make the result unusable.

2 Benchmark design

Let n be the particle count, u the per-frame update cost and r the render cost. The frame budget is

$$T(n) = u(n) + r(n), n \in 2^{14}, \dots, 2^{21} \quad (1)$$

The workload fixes the emitter geometry, the integration step and the blending mode, so that only the particle count varies between runs. The scene, the shader source and the build flags ship with the record.

Splitting the budget into update and render cost is the design decision that matters most. A single frame time hides whether a configuration is limited by simulation or by fill rate, and the two failures call for different fixes.

3 Measurement protocol

Each configuration reports the median frame time, the 95th percentile and the test device. Warm-up frames are discarded, the run is repeated three times, and every sample is archived

2^k	u (ms)	r (ms)	T (ms)
14	0.42	0.31	0.73
16	1.05	0.88	1.93
18	3.40	2.61	6.01
20	12.8	9.44	22.2

Table 1: Placeholder frame budget by particle count. Values demonstrate layout only.



Figure 1: Placeholder frame budget by particle count. Values demonstrate layout only.

next to the summary table. Reporting a central value together with a tail value keeps the discussion honest about stutter.

Every run records the thermal state before and after measurement. A benchmark that silently measures a throttled device is not wrong so much as unlabelled, and the record treats an unlabelled measurement as a failed one.

- Scene file, shader source and build flags published with the PDF.
- Device model, driver version and thermal state recorded per run.
- Raw frame-time samples archived next to the summary table.

4 Placeholder results

Table 1 shows the shape of the result table. Every value is a placeholder used to check column widths and caption placement.

Figure 1 draws the last column. The growth is super-linear, which is the expected shape for this workload and the reason the protocol reports a tail value as well as a median.

The placeholder numbers roughly double between adjacent rows for the smallest configurations and grow faster at the top of the range. That shape is typical of a fill-rate-bound workload once the particle count exceeds the point at which overdraw stops being hidden by the depth test.

5 Threats to validity

Three failures would invalidate a run without producing an error message: a background process competing for the GPU, a driver update applied between repetitions, and a scene file that differs in a single blending parameter. The protocol names all three so that a reader can check the archived files against them.

A fourth threat is the choice of particle counts itself. Powers of two make a clean figure and a misleading regression; a completed study would report a denser sweep and say how the counts were chosen.

6 Conclusion

A finished paper would add the analysis of the raw samples; the specimen keeps the structure visible and the measurements clearly marked as illustrations. The useful part of a benchmark is the part someone else can run.

The record therefore ends where it began: with the files. A reader who can rebuild the scene and the shader has everything the text promises, and a reader who cannot is reading a claim rather than a measurement.

References

- [1] Akenine-Möller, T., Haines, E., and Hoffman, N. (2018). *Real-Time Rendering* (4th ed.). CRC Press.
- [2] Pharr, M., Jakob, W., and Humphreys, G. (2023). *Physically Based Rendering: From Theory to Implementation* (4th ed.). MIT Press.
- [3] Hoefler, T., and Belli, R. (2015). Scientific benchmarking of parallel computing systems. In *Proceedings of SC '15*, 1–12.