GPU Computing

Exercise Sheet 07

Maximilian Richter, Jan Kränzke, Markus Everling,
Nicolas Schledorn, Olga Sergeyeva, Florian Feltz

January 6, 2025

1 Naive GPU Implementation

Using the naive GPU implementation, the performance increases roughly linearly with the number of bodies. This is a huge benefit compared to serial computation, where the performance scales with $\mathcal{O}(n^2)$. For 1024 particles, the simulation needs 55.5 ms to execute 100 iterations.

2 Optimized GPU Implementation

We prefer SOA over AOS, as it enables coalesced memory access and thus increased performance due to coherent memory layout. Using tiling and shared memory further increased the performance of the simulation. However, loop unrolling did not yield better performance in our case. This is probably due to overuse of registers.

Altogether, for 1024 particles, the simulation needs 9.7 ms to execute 100 iterations, where we found a block dim of 32 to be optimal.

3 n-Body GPU computations and streaming

The two implementations (with and without streaming) are basically the same, except that in the case of streaming, the memory has to be dynamically allocated over another loop. This increases the required computation time but enables particle numbers which exceed the graphics memory of the GPU.

The optimal number of streams is determined by the size of the GPU memory and the desired number of particles. The optimal number of streams is then chosen so that in a single stream the maximum number of particles which can be stored in memory is achieved.

4 Willingness to present

- 7.1 - **true**
- 7.2 - **true**
- 7.3 - **true**