Smoothed-Particle Hydrodynamics

GPU - Semestral project report

Ondřej Mézl

Problem description

Implementation of 3D Smoothed-Particle Hydrodynamics method for physics fluid simulation, on CPU and GPU, optimizing data structure and visualization. The SPH method works on particle level. Each step, particles add up influences of their neighbors in certain distance. Complexity of such operation is quadratic, therefore the particles are places in a grid to speed up the query in the average case.

Problem solution (Common for CPU and GPU)

The solution simulates fluid particles in a closed axis aligned bounding box.

The particles are generated on random positions with random velocities.

Particles are placed in a grid, so that a query for neighbors in certain distance does not (usually) require iterating all particles.

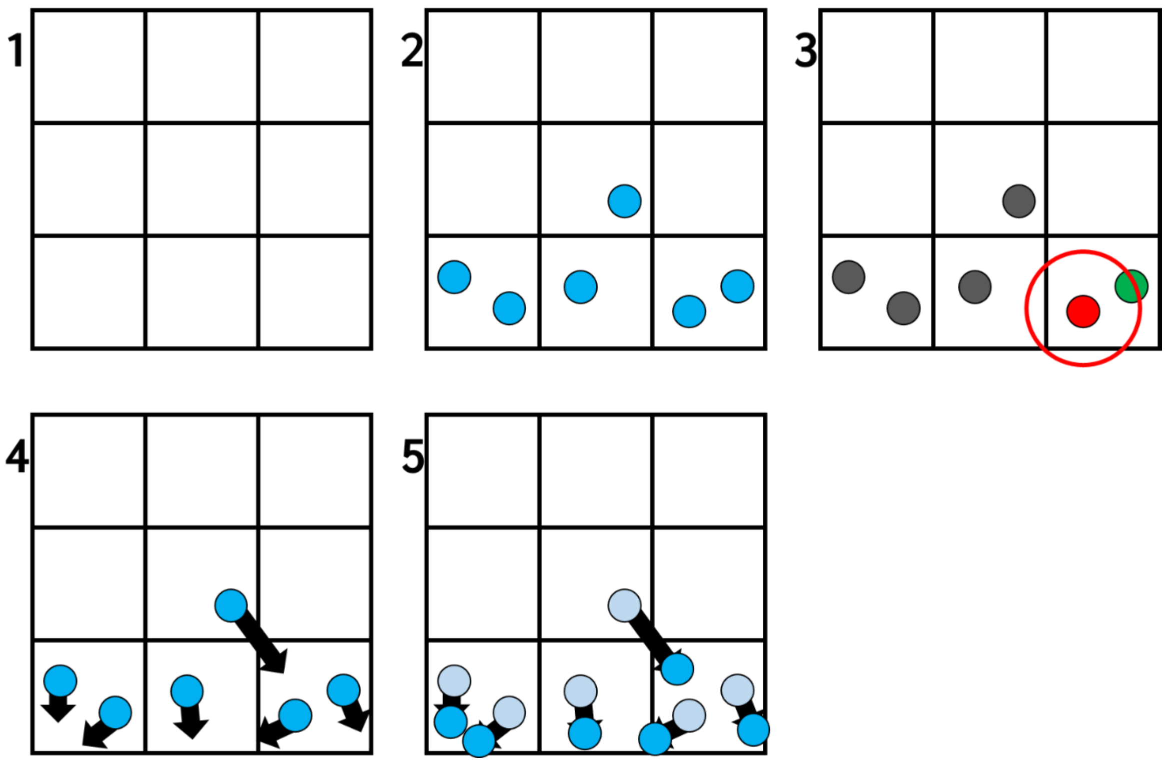
Every frame, simulation is updated with parameter of delta time, which is time last frame took.

This time is split into fixed number of steps, for each step calculation is executed. Reasons:

1. The simulation is more stable when it is computed at smaller time steps.
2. The simulation update is very fast and most of the time is spent of rendering, which is not the point of this task.

The **calculation step** is split into several parts:

1. Clear the grid
2. Fill the grid with particles (based on their position)
3. For each particle, find their neighbors and calculate density and pressure.
4. Based on distances among neighbors, densities, pressures, gravity (and various physics constants) calculate force.
5. Calculate velocities, update positions



Problem solution on CPU

CPU is always the director of the simulation; it loads the config, various resources and runs the simulation.

The simulation runs in parallel very easily: The calculation step mentioned above is executed in a thread pool and there is a barrier after every step.

The only exception is the neighbor buffer which uses 1 atomic variable as a counter, other than that there is no other synchronization needed.

Problem solution on GPU

The approach is exactly the same as CPU solution. CPU is still directing the simulation, the calculation steps are done on GPU, they are the same, but some implementation details are different (viz. below).

Visualization is done via OpenGL with GLFW and glad.

Implementation details (CUDA)

Thread count is equal to particle count, block size is 512. Grid and blocks are 1D.

For each calculation step (mentioned above) from 2., a kernel is executed. The step 1. (clearing the grid) is done by copying precomputed clear grid to a memory of a grid that is currently in use (device to device).

After all the steps are done, the state of particles is copied into host's memory (only once per frame).

Design and description of testing

Since this project is relatively unpredictable physics simulation that is initialized randomly, empirical tests were done to confirm the particles visually behave like fluid.

I have focused on these factors:

1. The particles eventually get into a stable, relaxed state (the simulation does not stay volatile and chaotic forever when reasonably set up).
2. The particles visually behave like a fluid would (waves, droplets, viscosity, fills up space).

Measurement results

The measurements were done separately on CPU and GPU, for 1 - 5000 particles. Only calculation time was measured, not copying data, rendering or anything else. I have measured time to perform 100 calculation steps.

Since the times were a bit unstable based on where the particles were (the grid data structure works better if the particles are spread, worse if they are closely together), I have waited for the simulation to get more stable and took average time. Therefore, I should get results that are close to the worst case, since in a stable situation, particles are usually laying closely together on the ground.

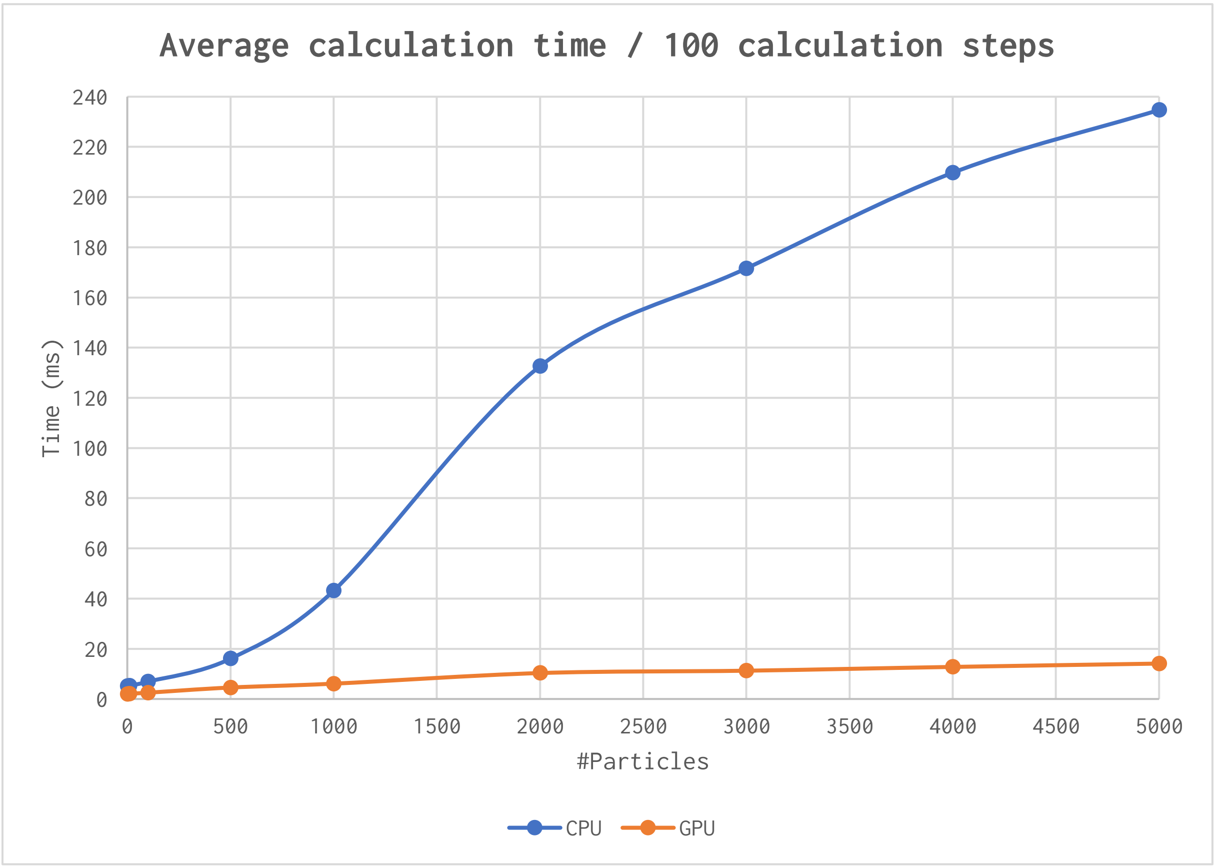
The measurements were done on this hardware:

CPU: AMD Ryzen 5 1600 Six-Core Processor (12 CPUs, ~3.2GHz)

GPU: NVIDIA GeForce GTX 1660 SUPER

PARTICLES	CPU	GPU	SPEEDUP
1	5.2	1.97	2.6
10	5.2	2.07	2.5
100	6.9	2.50	2.8
500	16.1	4.57	3.5
1000	43.1	6.10	7.1
2000	132.6	10.33	12.8
3000	171.6	11.27	15.2
4000	209.7	12.78	16.4
5000	234.7	14.11	16.6

This table contains these columns: Number of particles, CPU time (ms / 100 calculation steps), GPU time (ms / 100 calculation steps), Speed-up



Results evaluation

A significant speed-up was measured, GPU simulation is about 16x faster than the CPU one.

Conclusion

Smoothed-particle hydrodynamics, a method for fluid simulation was implemented in a parallel manner, on CPU and GPU.

An optimizing data structure (grid) was used.

Correctness was evaluated empirically based on visual behavioral resemblance to fluids.

A significant speed-up, about 16x, was observed in GPU variation, compared to the CPU one.