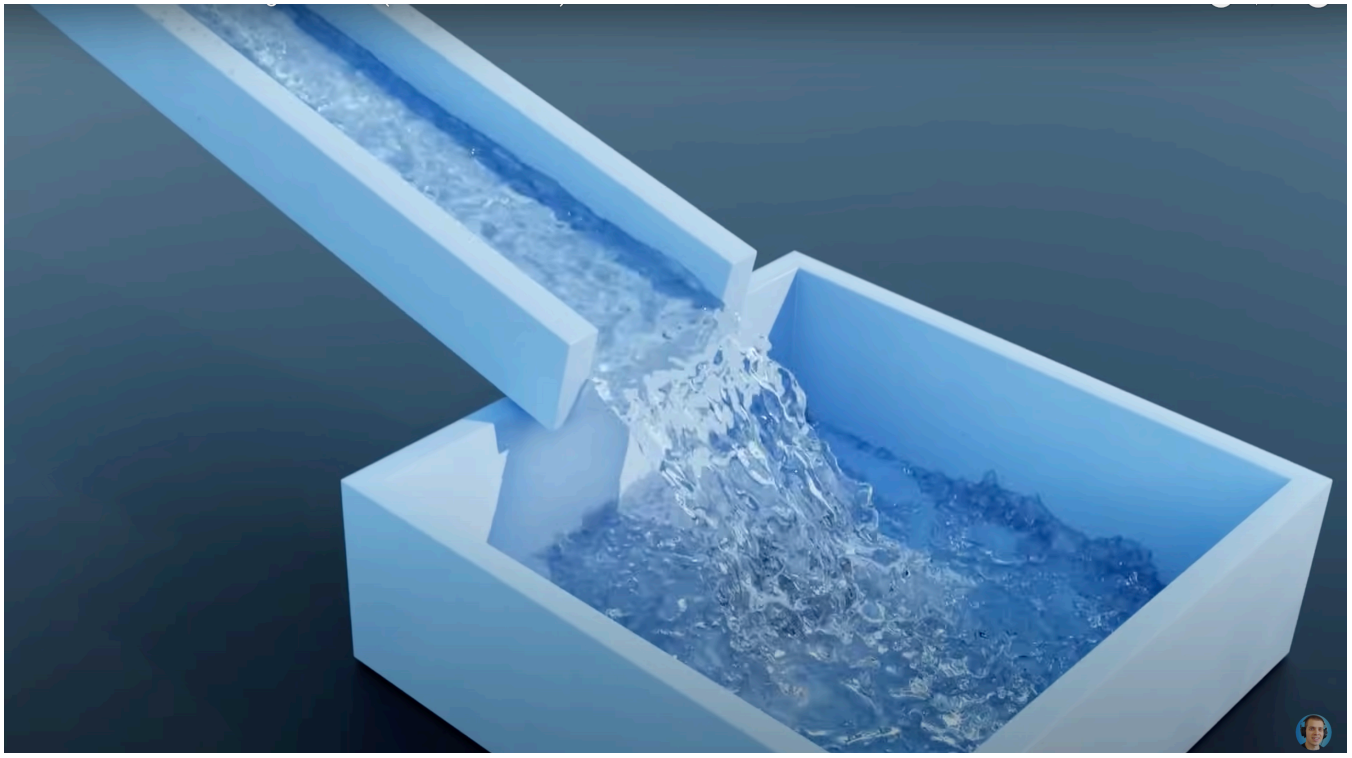
Parallel Smoothed Particle Hydrodynamics Fluid Solver

Team Members: Zhixin Chen, Wencheng Zhang

Website: https://wcvanvan.github.io/SPH-CUDA-Web/

SUMMARY

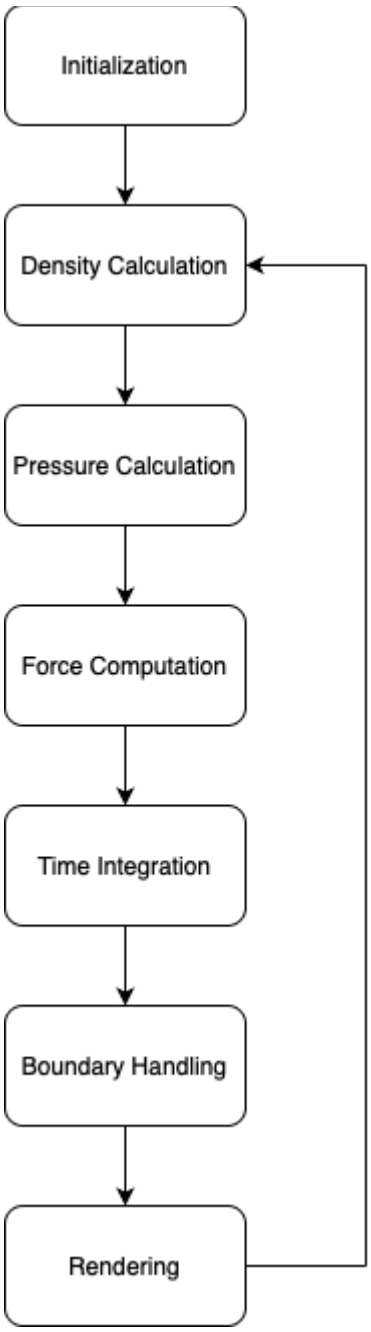
We are going to implement an optimized Smoothed Particle Hydrodynamics (SPH) fluid solver on the NVIDIA GPUs in the lab. Specifically we are simulating water flowing down the trough into a sink as shown in the image below.



BACKGROUND

In SPH, fluids are represented by particles, and each particle carries properties such as mass, position, velocity and so on. The fundamental idea is to estimate fluid properties at any point by averaging the contributions from neighboring particles.

A block diagram of SPH is shown below. Every particle do pressure calculation, force computation, and time integration independently, so parallelism can be fully exploited in these steps.



THE CHALLENGE

1. NEIGHBOR SEARCH IN IRREGULAR 3D SPACE: The fundamental challenge lies in efficiently finding neighboring particles within the smoothing radius for each particle. In dynamic simulations, particles move freely, creating constantly changing neighborhoods that can't be predicted in advance. A brute-force approach of checking every particle against every other particle results in $O(N^2)$ complexity, which becomes expensive as particle counts increase into the thousands.
2. COMMUNICATION AND DATA DEPENDENCIES: Density calculation requires each particle to gather position data from all its neighbors, creating significant communication overhead. This data dependency means particles cannot be processed independently, leading to potential bottlenecks in parallel implementations. The irregular memory access patterns also hurt cache performance, further reducing computational efficiency.
3. KERNEL EVALUATION OVERHEAD: Each neighbor interaction requires evaluating complex smoothing kernel functions involving square roots and high-order polynomials. These mathematical operations are computationally expensive when performed thousands of times per simulation step.

4. LOAD BALANCING: In parallel computing environments, the varying number of neighbors per particle creates significant load imbalance. Some threads finish quickly while others lag behind, reducing overall utilization of computational resources. This thread divergence is particularly problematic on GPU architectures designed for uniform workloads.

RESOURCES

- Computers: GHC Computers with NVIDIA GPUs
- Starter code: Start from scratch
- References: Müller, Matthias & Charypar, David & Gross, Markus. (2003). Particle-Based Fluid Simulation for Interactive Applications. Fluid Dynamics. 2003. 154-159.

GOALS AND DELIVERABLES

Plan to achieve: Simulate water flowing down the trough into the sink

Hope to achieve: Turn the trough into a curved pipe which is a more complex geometry

Fallback goals: Only simulate water flowing in a rectangular sink

Poster session targets: Run real-time water simulation demo

PLATFORM CHOICE

In fluid simulation, thousands of particles are required as a minimum for a decent demo. Nvidia RTX 2080 can easily handle thousands of concurrent threads, which perfectly suits our needs.

SCHEDULE

- **Week 1 (03/26 - 03/30):** Read the reference paper and start coding
- **Week 2 (03/31 - 04/06):** Run demo showing water flowing in a rectangular sink
- **Week 3 (04/07 - 04/13):** Run demo showing water flowing down a trough, and finish project milestone report
- **Week 4 (04/14 - 04/20):** Optimize density calculation step
- **Week 5 (04/21 - 04/27):** Final preparations and testing