15418 Final Project: Fluid Simulation Exploration

Kevin You
Katerina Nikiforova

April 2025

1 Summary

This project will explore a parallel, highly optimized hybrid Lagrangian/Eulerian fluid simulation, focusing on real-time performance. We plan to explore both the modern standard of Fluid Implicit Particle (FLIP) fluid simulation methods and the more recent Affine Particle-In-Cell (APIC) that attempts to alleviate the potential instabilities of the FLIP method.

2 URL

https://kn1451j.github.io/ParallelFluidSim/

3 Background

Hybrid Lagrangian/Eulerian methods combine the advantages of Lagrangian (particle-based) methods and Eulerian (grid-based) methods. To use a hybrid method, it is necessary to store simulation data, that is, mass, velocity, and positions, on Lagrangian particles that move around the scene, and transfer these data to the grid cells they are contained in perform the global pressure projection on the grid, and transfer the data back to the particles. This is done using the Navier-Stokes equations.

The Navier-Stokes equation describes the interaction of a volume under pressure and external forces:

$$\rho\frac{dv}{dt} + \rho\Delta v \cdot v = -\Delta p + pg \tag{1}$$

In this project, we also assume that the fluids we simulate will be incompressible fluids, which includes water, and implies zero total divergence.

$$\Delta \cdot v = 0 \tag{2}$$

Using the above two equations, we can derive phase-based update steps for incremental particle updates. The implementation of these updates varies between FLIP and APIC methods, which in particular we hope to explore in their capabilities for parallelization and work splitting between individual particles.

3.1 Fluid Implicit Particle Implementation

We plan to implement a hierarchical and parallelized version of FLIP for GPU acceleration. FLIP is a particular simulation algorithm of the Navier-Stokes equations, which discretizes space into a grid. The algorithm follows as:

1. Apply incremental per-particle update steps for the velocities based over a timestep, solving for velocity change under advection and external forces
2. Project particles into the grid
3. Solve a partial differential equation linearized into the pressure projection equation $Ap = d$
4. For each grid cell take, $v^{t+1} = v^B - \frac{\Delta t}{\rho} \Delta p$
5. Apply a weighted transfer of velocity to the particles by transferring change in grid velocity: $\Delta v(j) = v^{t+1}(j) - v^t(j)$

In particular, we will re-implement the hierarchical voxelization explored in [**fastFLIP**] and profile its CUDA re-implementation. We will compare the performance of this implementation in relation to the APIC method described below.

In particular, we plan to explore how a sparse grid structure can be distributed across a combination of CUDA threads most efficiently. We hope to explore different methods of "tiling" the grid across threads, including 3D blocking across x,y, and z axes.

3.2 Affine Particle in Cell Implementation

4 Challenges

- Fluid simulation generally poses a problem for parallelization due to the interaction between particles and creates communication overhead between grid cells. However, by breaking down the simulation into timesteps, we hope to alleviate this contention.
- The nature of individual particles allows for parallelization, but typical considerations like workload balance and locality must be considered. Locality of particles is a relevant problem, and we hope to explore different ways of dividing our 3D grid representations across threads.

- FLIP and APIC will differ in amount of variables needed on particles and grid nodes which will likely be significant when considering the size of cache lines.
- A significant bottleneck of the simulation is the global pressure projection phase. We aim to simulate free-surface fluids, where significant regions of the simulation domain may be empty, so we aim to speed up the conjugate gradient solver via sparse data structures.

5 Resources

We will experiment on a machine with an Intel Core i9-14900KF with 32 GB memory and an NVIDIA GeForce RTX 4080.

6 Deliverables

Our minimum deliverable will be a parallelized FLIP simulation with optimized particle-grid transfers. Extra goals include also implementing APIC and improving the conjugate gradient solver.

We plan to use a pre-built renderer for particles to visualize our results. We will implement profiling for each of the implementations and compare their speeds in across both the grid and particle phases of the simulation.

We will experiment with both implementations' CUDA parallelization capabilities and present quantitative comparisons on the speedup between the sequential and parallel runtime of both the APIC and FLIP implementations.

7 Schedule

- April 8th - Complete implementation of FLIP and APIC on Taichi with no special optimizations
- April 15th - Implementations of FLIP and APIC in C++ with CUDA
- April 21st - Instrument real-time visualization and profiling tools for comparing performances
- April 28th - Finish conducting experiments, write final report