Physically-Based Simulation Final Presentation: Dam Overflow

Group 21
Ryan Ammann, Paul Ochs

Fluid Implicit Particle Method

- Hybrid Method
- Uses grid based computation for solving pressure equations to keep the simulation incompressible
- Uses particle based computation for “advection” since this is way cheaper than for fully grid based

$$u_p^{new} = \underbrace{\alpha * lerp(u_{grid}^{new}, x_p)}_{\text{PIC contribution}} + \underbrace{(1-\alpha)[u_p^{old} + lerp(\Delta u_{grid}, x_p)]}_{\text{Flip contribution}}$$

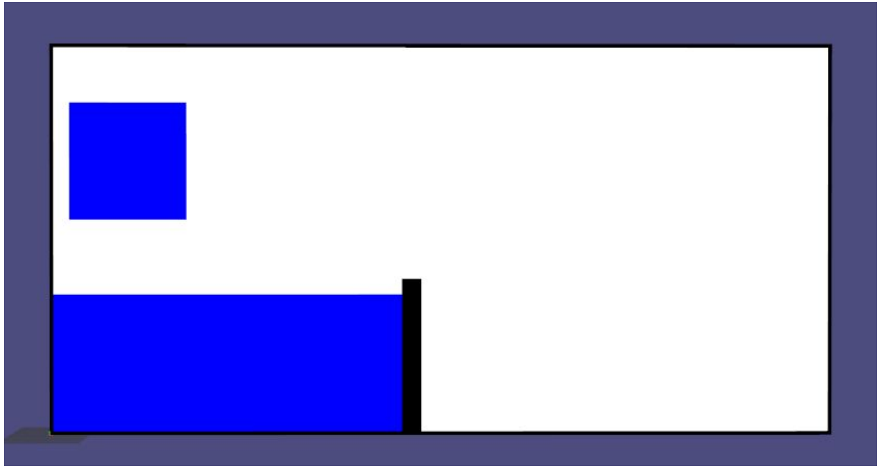
One Simulation Step

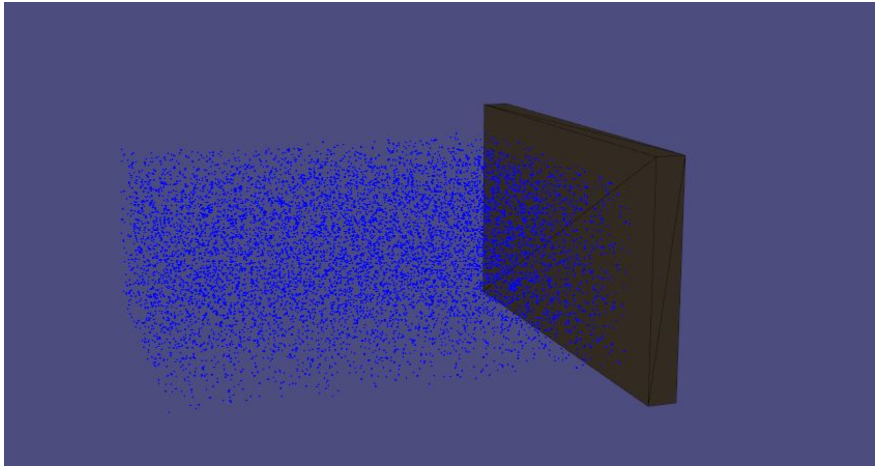
Do particle to grid transfer	→	`compute_velocityfield();` `save_velocities();`

Add forces and boundaries	→	`add_forces();`
		`add_boundries();`
		`classify_cells();`

Solve pressure	→	`solve_poisson();` `apply_pressure();`
Do grid to particle transfer	→	`update_particle_velocities();`
Get suitable timestep	→	`double stable_dt = cfl_timestep();` `int substeps = std::ceil(m_dt / stable_dt);`
Advect particles	→	`for (int i = 0; i < substeps; i++)` `    advect_particles(stable_dt);`

Implementation





Stability and Optimization

- 2D implementation fairly stable, assuming “correct” damping and pressure
- 3D implementation more volatile
- Adaptive time-stepping

$$u_p^{new} = \alpha * lerp(u_{grid}^{new}, x_p) + (1 - \alpha)[u_p^{old} + lerp(\Delta u_{grid}, x_p)]$$

- Some small attempts at parallelization using OpenMP
- Simple compiler flags (-O3 , march = native …) yielded best speedup

In Conclusion